INTERNATIONAL JOURNAL OF SPEECH TECHNOLOGY 5, 355–370, 2002

Large Vocabulary Search Space Reduction Employing Directed Acyclic Word Graphs and Phonological Rules

KALLIRROI GEORGILA, NIKOS FAKOTAKIS AND GEORGE KOKKINAKIS
Wire Communications Laboratory, Electrical and Computer Engineering Department, University of Patras, Greece
rgeorgil@wcl.ee.upatras.gr
fakotaki@wcl.ee.upatras.gr
gkokkin@wcl.ee.upatras.gr

Abstract. Some applications of speech recognition, such as automatic directory information services, require very large vocabularies. In this paper, we focus on the task of recognizing surnames in an Interactive telephone-based Directory Assistance Services (IDAS) system, which supersedes other large vocabulary applications in terms of complexity and vocabulary size. We present a method for building compact networks in order to reduce the search space in very large vocabularies using Directed Acyclic Word Graphs (DAWGs). Furthermore, trees, graphs and full-forms (whole words with no merging of nodes) are compared in a straightforward way under the same conditions, using the same decoder and the same vocabularies. Experimental results showed that, as we move from full-form lexicons to trees and then to graphs, the size of the recognition network is reduced, as is the recognition time. However, recognition accuracy is retained since the same phoneme combinations are involved. Subsequently, we refine the *N*-best hypotheses' list provided by the speech recognizer by applying context-dependent phonological rules. Thus, a small number *N* in the *N*-best hypotheses' list produces multiple solutions sufficient to retain high accuracy and at the same time achieve real-time response. Recognition tests with a vocabulary of 88,000 surnames that correspond to 123,313 distinct pronunciations proved the efficiency of the approach. For $N = 3$ (a value that ensures we have fast performance), before the application of rules the recognition accuracy was 70.27%. After applying phonological rules the recognition performance rose to 86.75%.

Keywords: large vocabulary speech recognition, automatic directory assistance services, trees, Directed Acyclic Word Graphs (DAWGs), search space reduction, context-dependent phonological rules

1. Introduction

The automation of Directory Assistance Services (DAS) has attracted great interest in the last decade due to the visible benefits both for the telephone companies and the subscribers. For example, every year telephone companies in the United States spend over $1.5 B providing DAS. Typically it takes the operator about 25 sec to complete a DAS call. A reduction of only one second in this average work time represents a savings of over $60 M a year (Lennig et al., 1995). On the other hand, customers benefit from the fact that they are served without delays and far beyond working hours, possibly 24 hours a day.

Several demonstrations have been reported, such as the system of British Telecom (Whittaker and Attwater, 1995), FAUST (Kaspar et al., 1995) and PADIS-XL (Seide and Kellner, 1997). PADIS (Philips Automatic Directory Information System) has a system driven dialogue where the caller must reply with only one word, spelled or spoken, per dialogue turn, and handles a database of 131,000 entries. Recently a system based on PADIS, which can handle a complete country has been presented in Schramm et al. (2000). Nortel

has deployed its product ADAS Plus (Automated Directory Assistance System-Plus), which partially automates the DAS function through speech recognition, in Quebec. This system distinguishes between two languages (English and French) and automates the recognition of city names (Gupta et al., 1998). In Italy, Telecom Italia carried out in July 1998 a field trial in 13 districts using a system designed to completely automate a portion of calls on a country wide basis. This implies recognition of about 25 million directory entries distributed in 8,105 towns. The required parameters are collected separately through specific requests to the user. They are supposed to be uttered in isolation, e.g., "Torino", not "the city of Torino" (Billi et al., 1998). The Durham telephone enquiry system has been successfully applied to English and Italian telephone databases of up to 100,000 entries (Collingham et al., 1997). The DirectoryAssistant of Phonetic Systems (http://www.phoneticsystems.com) utilizes a patented core technology of advanced probability-based algorithms to perform sophisticated searches of extremely large databases. It is currently commercially deployed delivering speech-enabled DAS for over 5 million wireline and wireless telephone listings in Finland, in cooperation with Sonera Info Communications Ltd.

In this paper we present a spoken dialogue system for automating DAS that was developed in the framework of the EU project IDAS[1] and then extended and improved so that it can be utilized in real-world conditions. Another demonstration also funded by IDAS has been reported in Córdoba et al. (2001). In automatic directory information systems, a speech recognizer is expected to be able to handle very large vocabularies. Moreover, these vocabularies are expected to be "open-set lexicons" meaning that more words, e.g., surnames, first names, city names, may need to be added later. Therefore, efficient techniques able to cope with the above constraints should provide a real-time search operation over the whole vocabulary structure and a means of easy vocabulary augmentation, and at the same time give high accuracy rates. The greatest part of this paper focuses on the algorithms developed to handle large vocabulary recognition issues. However, the dialogue flow is also described to give the reader an overall picture of the application and its special features.

The paper is organized as follows: Section 2 presents an overview of the system. The techniques applied to deal with search space reduction issues are described in detail in Section 3. The performed experiments are presented in Section 4. Finally, a summary and conclusions are given in Section 5.

2. System Overview

2.1. Dialogue Strategy

In the first step of the dialogue the system asks the user if s/he is looking for the telephone number of a company, an organization/institute or a person. A typical dialogue in which the caller requests the telephone number of a company or organization/institute is as follows:

...
System: *Please give the city name.*
Caller: *The city is Athens.*
System: *Could you please specify the district?*
Caller: *The organization is located in Kallithea.*
System: *Please give the name of the organization.*
Caller: *Greek Organization of Tourism.*
System: *The number you requested is ...*

If the user gives the city name of Athens or Thessaloniki (the biggest cities in Greece), the system will prompt him/her to specify a district in the above city. However, the caller could also give directly the name of the district, without having to utter the city name first. In those cases in which the system cannot find the requested telephone number in the district provided by the caller, it will extend the search space to the other districts of the city as well. Thus, it is ensured that even if the user has no knowledge about the exact district, which happens very often, s/he will be able to get the desired information.

Figure 1 depicts the dialogue flow in case the user requests a person's telephone number. A typical dialogue is as follows:

...
System: *Please give the city name.*
Caller: *Patras.*
System: *Please utter the first letter of the surname.*
Caller: *It starts with a G.*
System: *Please give the person's surname.*
Caller: *His name is Georgiou.*
System: *Please give the forename of the person.*
Caller: *Alexis.*
System: *The number you requested is ...*

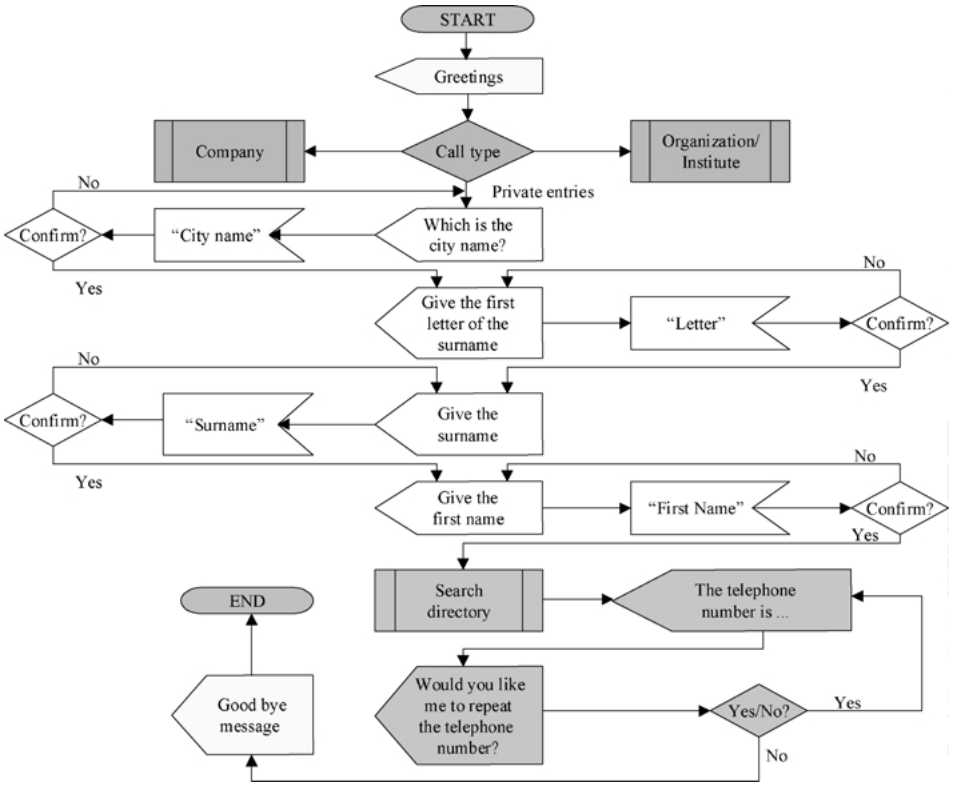

Figure 1. Dialogue flow of the system.

The above example shows that the system asks for the first letter of the person's surname in order to reduce the search space because the surname recognition involves far two many candidate solutions compared to the recognition of companies or organizations/institutes. After the system has gathered the necessary information, it searches the telephone directory, and the telephone number asked for is spoken to the user as a mixture of prerecorded speech (for the prompt) and synthesized speech (for the digits that form the telephone number). If the search in the database produces more than one solution, the system will inform the user about all of them.

2.2. *Comparison with Other Approaches*

An efficient search through a large vocabulary structure may be performed by two common methods: the first is to reduce the size of the active vocabulary in every dialogue turn and the second to use spelling.

In the Philips Automatic Directory Information System (Seide and Kellner, 1997), the dialogue flow is as follows: 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. Note that here the caller must utter only one word per dialogue turn, e.g., "Aachen", whereas in our system there is no such restriction. That is, the utterance "he lives in Athens" is allowed and will be correctly processed.

In the British Telecom Automatic Directory Assistance Service (Whittaker and Attwater, 1995) the dialogue model is somewhat 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. In this case the recognition task is more difficult because the entire vocabulary is active. Therefore, if the recognizer provides a solution with high probability then the recognition result is almost certain to be correct, which implies a high value of confidence. On the other hand, if successive recognitions are constrained by previous ones then the recognition task is easier since the active vocabulary is restricted. Thus, accuracy gets higher and confidence decreases.

Phonetic Systems employs either search space reduction with every dialogue turn or a method of searching the entire dictionary using a path that passes through the words with the highest probability of being correct. In the second case, the entire dictionary is organized in such a way as to examine the input word against various basic characteristics of each word in the restructured dictionary subset. Only those words that "pass" these preliminary checks will be further compared with the input word using more extensive and sophisticated similarity checks (Phonetic Systems, 2002).

In our system each dialogue turn is independent of the previous ones. Therefore the search space is not reduced with every dialogue turn, with only two exceptions. The first one is between the subsequent dialogue turns of prompting the caller to give the first letter of the surname and then fully utter it. In this case, the search space is reduced significantly since now the active vocabulary consists only of the surnames that start with the previously recognized letter. The second exception is between the dialogue turns of asking for the city name and then for a specific district (only for Athens and Thessaloniki). Now the active vocabulary is restricted to the districts of the previously selected city.

The reason we have decided to keep dialogue turns independent of each other is that we are interested in high confidence. Nevertheless, experimentation with constrained recognitions by previous ones is a process in progress, which requires that the speech recognizer be improved so that possible recognition errors do not affect the subsequent dialogue turns. An additional reason for the independence of dialogue turns is that it deals with the problem that would arise otherwise if the caller gave a false district. If the search space was reduced with every dialogue turn and the system failed to find the requested information in the district specified by the user, it would not have the alternative solution of extending the search to other districts in the same city. This is because the list of active surnames or first names would have been limited to include only surnames and first names of the selected district.

In Greek, spelling is not usual (splitting the word in syllables is preferred), and thus we have decided not to use it in our dialogue system. In our application, the EU project IDAS, the recognizer must distinguish between 257,198 distinct surnames that correspond to 5,303,441 entries in the directory of the Greek Telephone Company. By restricting the search space to the most frequent 88,000 ones that correspond to about 123,313 distinct pronunciations, 93.57% of the directory's listings is covered. Kamm et al. (1995) performed a study on the relationship between recognition accuracy and directory size for complete name recognition and reached the conclusion that accuracy decreases linearly with logarithmic increases in directory size. The above conclusion shows that it is necessary to develop efficient algorithms for handling large vocabulary recognition issues. Although the motivation behind their development was the lack of the use of spelling in Greek, the techniques that will be described in the following sections are suitable for any language.

2.3. Speech Recognition

The speech recognizer we use was built with the HTK Hidden Markov Models toolkit (Young et al., 1997), which is based on the Frame Synchronous Viterbi Beam Search algorithm. The acoustic models are tied state context-dependent triphones of five states each. In order to train the recognizer we used the SpeechDat-II Greek telephone database (Van den Heuvel et al., 2001). This database is a collection of Greek annotated speech data from 5000 speakers (each individual having a 12-minute session). We made use of utterances taken from 3000 speakers in order to train our system.

In all dialogue turns, the HTK decoder traverses a word network containing possible speaker utterances in order to find the N-best hypotheses. The candidate solutions, e.g., surnames, first names, city names, form a sub-network of the full network. In surname recognition in order to deal with the large vocabulary issue, we replace the word sub-networks of surnames with phoneme networks that can produce the phonetic transcriptions of all the above surnames using DAWG (Directed Acyclic Word Graph) structures. A DAWG is a special case of a finite-state automaton where no loops (cycles) are allowed. DAWGs allow sharing phones

across different words (as opposed to using a separate instance for every phone in the pronunciation of each word), which reduces recognition search space and therefore response time. Most speech recognition systems that have to deal with very large vocabularies use a tree structure (i.e., trie) (Gopalakrisnan et al., 1995; Nguyen and Schwartz, 1999; Suontausta et al., 2000). However, trees are not the optimal way to represent lexicons, due to their inadequacy to exploit common word suffixes. For this reason, the use of DAWG structures is more appropriate. DAWGs have been successfully used for storing large vocabularies in speech recognition. Hanazawa et al. (1997) used an incremental method (Aoe et al., 1993) to generate deterministic DAWGs. The aforementioned method was applied to a 4000-word vocabulary in a telephone directory assistance system. However, in Hanazawa et al. (1997) the comparison between the tree and the DAWG was made using different decoding algorithms. Thus the efficiency of the DAWG was not shown under the same conditions. Betz and Hild (1995) used a minimal graph to constrain the search space of a spelled letter recognizer. However, neither did they report details on the algorithm they applied, nor did they compare graphs with full-forms (whole words with no merging of nodes) and trees. Our novelty is the comparison between full-forms, trees and graphs under the same conditions, that is, using the same decoder and the same vocabularies. Furthermore, we use trees and graphs with a conventional decoder in contrast with other existing techniques (Hanazawa et al., 1997; Suontausta et al., 2000).

Since there is no dialogue turn for spelling and the caller is prompted directly to utter the surname, the value of N in the N-best hypotheses' list of the speech recognizer must be high. This will 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 will slow down the system's response.

After the speech recognizer has produced the N-best hypotheses, context-dependent phonological rules are applied, which define classes of phonemes and phoneme combinations, the members of which can be falsely recognized in a specific context. That is, a phoneme or phoneme combination of a class could be mistaken for another phoneme or phoneme combination of the same class in the context defined by the rule. Thus, recognition errors and pronunciation variability are taken into consideration. The solutions created by applying the phonological 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 phonemes or phoneme combinations, but without leading to too many solutions. On the other hand, the rules' processing algorithm is language-independent.

Most approaches incorporate pronunciation variation into the lexicon that will be used by the recognizer in the decoding process (Chen, 1990; Schmid et al., 1993; Ramabhadran et al., 1998). Our proposal is to apply information on pronunciation variation in a separate stage after the recognition task. That is, we apply phonological rules to the recognizer's output. The advantage of such an approach is the gain in response time. The cost of processing the signal in order to produce multiple outputs is much higher than the time required for taking an output and applying the phonological rules.

A similar approach has been applied to letter recognition in Mitchell and Setlur (1999). 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 in which 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 with a tightly constrained grammar.

3. Search Space Reduction Techniques

3.1. Construction of Full-Forms, Trees and Graphs

Our approach to the use of DAWGs for large vocabulary speech recognition was first described in Georgila et al. (2000). In the current work, we explain this technique in more detail, and we show how our method can be used in conjunction with phonological rules for achieving high accuracy rates. Furthermore, the above techniques are implemented in a real-world application.

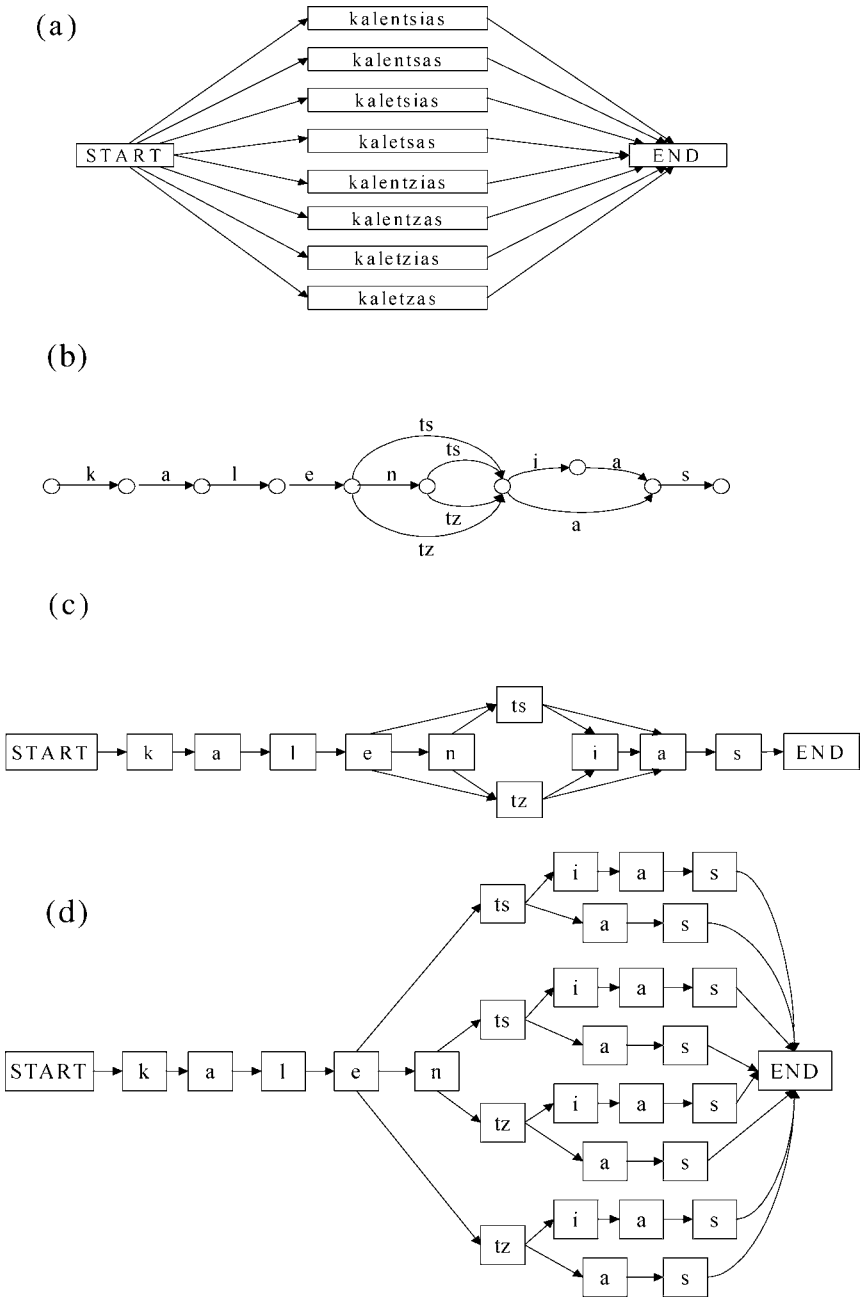

Figure 2. (a) Full-form word network, (b) phoneme DAWG produced by our incremental algorithm, (c) phoneme graph in the decoder format, and (d) phoneme tree also in the decoder format.

We use incremental construction of DAWGs in order to be able to update them without having to build them from scratch in every change. We have used the incremental construction algorithms described in Sgarbas et al. (1995, 2001) because we are particularly interested in non-deterministic DAWGs since they appear to be even more compact than the minimal deterministic ones (Sgarbas et al., 2001).

A word (full-form) network consisting of surnames is replaced by a phoneme network that can produce the phonetic transcriptions of all the above surnames (Fig. 2(a) and (c)). Thus, a lexicon of surnames in

phonetic transcription (Fig. 2(a)), is first transformed into a Directed Acyclic Word Graph (DAWG) (Fig. 2(b)). Our algorithm produces the DAWG of Fig. 2(b), where simple monophone pronunciations label the transitions between nodes. The next stage of the method is to convert these structures into the format that is accepted by the HTK decoder (see Section 3.3), where the labels are on the nodes (Fig. 2(c)). Finally, the tree of Fig. 2(d), also in the HTK format, is derived from the graph.

If the surnames in Fig. 2 had multiple pronunciations, they would be treated as different words by the algorithm. Using the above network reduction method, we get an equivalent but more compact network, which results in faster search. In both the tree and the graph several words have a common path, thus recognition is substantially accelerated in comparison to the full-form network, when the same recognizer is used in all networks. Furthermore the graph is more compact than the tree since common suffixes are also merged.

3.2. *Structure of Phonological Rules*

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 $i = 1, \ldots, k$ is the left context of the rule, S is the class, which includes phonemes or phoneme combinations that could be interchanged, and $R_p p = 1, \ldots, n$ 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 phonemes or phoneme 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 at only one class of phonemes or phoneme combinations backwards and three forward. Nevertheless, the processing algorithm is parametric and could work for any values of k and n.

There are three types of rules: substitution, insertion and deletion rules. The following rule is a substitution rule in which g and k are interchanged ($k = 1$ and $n = 3$):

$$\text{-}, g\ k, \#w, NULL, NULL \qquad \text{(Rule 1)}$$

where *NULL* stands for any step not considered by the rule and the dash for an empty string. Rule 1 states that g can be interchanged with k, when no phoneme precedes them and when they are followed by any phoneme or phoneme combination contained in cluster $\#w$. The first character of a cluster symbol is always # to avoid conflicts when characters are used both as phonemes and cluster names. We are not interested in what follows after $\#w$ and that is what the 2 *NULL* symbols denote in Rule 1. Cluster $\#w$ is defined as

$$\#w = (a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, \text{-})$$

that is, $\#w$ includes all the letters of the English alphabet plus the dash. Note that not all letters are used as phoneme symbols but here we have included all of them in cluster $\#w$ to stress the fact that we are indifferent to what follows after g or k. The dash is used when we are at the beginning of a word's phonetic transcription (left context) or at the end (right context). The use of clusters prevents us from having too many rules, e.g., -, $g\ k$, a or -, $g\ k$, e etc.

In the same way we have the following rule:

$$\#w, tsi\ ts, \#w \qquad \text{(Rule 2a)}$$

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

$$NULL, \#w, tsi\ ts, \#w \qquad \text{(Rule 2b)}$$

or

$$\#w, ts, i\text{-}, \#w \qquad \text{(Rule 2c)}$$

Rule 2 may be considered as a deletion rule if we have *tsi* and we replace it with *ts* or as an insertion rule in the opposite case, that is when we have *ts* and we replace it with *tsi*. 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.

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

$$\#w\text{-}, r(\#v1)k\ rk, \#w \qquad \text{(Rule 3)}$$

where

$$\#w\text{-} = (a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z)$$

and

$$\#v1 = (a, e, i, o, u)$$

Rule 3 says that *rk* can be interchanged with *rak*, *rek*, *rik*, *rok*, or *ruk* in a specific context. Rule 3 is considered as deletion or insertion rule according to the input substring, which activates the rule. The left context is class #*w*-, which means that any phoneme could precede a phoneme or phoneme 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's phonetic form. The right context is class #*w*, which means that any phoneme can follow. Cluster #*v1* contains all the vowels. However, this leads to rare combinations. That is, *rak* cannot be mistaken for *rk* easily. 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 or rare solutions, which will lead to increasing the system's response time. In the same way cluster #*w* in Rule 1 leads to invalid combinations, e.g., *gp* 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 phoneme combinations. That is, if the designer tried to make clusters that would include only the appropriate (not redundant) phonemes or phoneme combinations for a specific context, it is very likely that s/he would fail to consider all the cases for this particular context.

Our rules contain both phonetic and linguistic knowledge. For example, in Rule 1 we use the phonetic knowledge that since *g* and *k* are both velar plosives they could replace one another. On the other hand, Rules 2 and 3 exploit the linguistic knowledge that some phonemes or phoneme combinations could be interchanged in a specific environment. Currently the rules are extracted manually. However, research in developing an algorithm for their automatic extraction is in progress. We aim at developing an algorithm for the automatic extraction of rules, which will exploit both the linguistic knowledge contained in phonetic transcriptions of words, and the information carried in the speech signal itself.

3.3. Decoding Process

As it has already been mentioned, the speech recognizer we use is the HTK Hidden Markov Models toolkit. All possible speaker utterances form a network, in which the nodes are words or sub-words or even single letters and the arcs represent the transitions between nodes. Given the set of the acoustic models (HMMs), the network and the corresponding dictionary, which contains the phonetic transcriptions of the words, subwords or letters that correspond to the nodes, HTK produces the N-best hypotheses.

In all our experiments we have used three different types of networks together with their corresponding dictionaries. In the first case the nodes are full surnames (Fig. 2(a)). The corresponding dictionary contains the monophone transcriptions of these words. Using the above dictionary, HTK expands the word network of Fig. 2(a) to the network of Fig. 3(a) during decoding. Each word in the word network is transformed into a word-end node preceded by the sequence of model nodes corresponding to the word's pronunciation as defined in the dictionary. Monophones are expanded to context-dependent triphones, and there is also cross-word context expansion between the *sp* (short pause) model of the *START* and *END* nodes and the models that form the full surnames. The second case refers to the graph-based phoneme network shown in Fig. 2(c), and the third to the tree-based phoneme network depicted in Fig. 2(d). Now the dictionary consists of *START* and *END* corresponding to *sp* and monophones having themselves as pronunciation. The network of Fig. 2(c) is expanded to the one in Fig. 3(b), and the network of Fig. 2(d) to the one in Fig. 3(c). If we compare the networks of Fig. 3(a)–(c) it is clear that the second and third networks are more compact and contain fewer model nodes than the first one. However, the number of word-end nodes increases since each monophone is considered as a distinct word. The conducted experiments described in Section 4 prove that as the size of the vocabulary increases the total number of nodes and links of an expanded phoneme network (tree or graph) is getting smaller than the one of an expanded word network. This is something expected because, in both cases, links are merged in order to produce an efficient network. Therefore, as word networks grow larger we will reach a point where their equivalent phoneme networks have fewer word-end nodes due to the merging process. In addition, if context-dependent triphones have been tied during training, their model nodes are merged. This could lead to further decrease of the phoneme network's size. It should be noticed that sometimes during the expansion *NULL* nodes must be inserted. But even if they increase the number of nodes and links in some cases, they do not add to the processing time, as explained in the following. Graph-based networks are more compact than the corresponding tree-based ones, because not only prefixes are merged as in trees, but also suffixes.

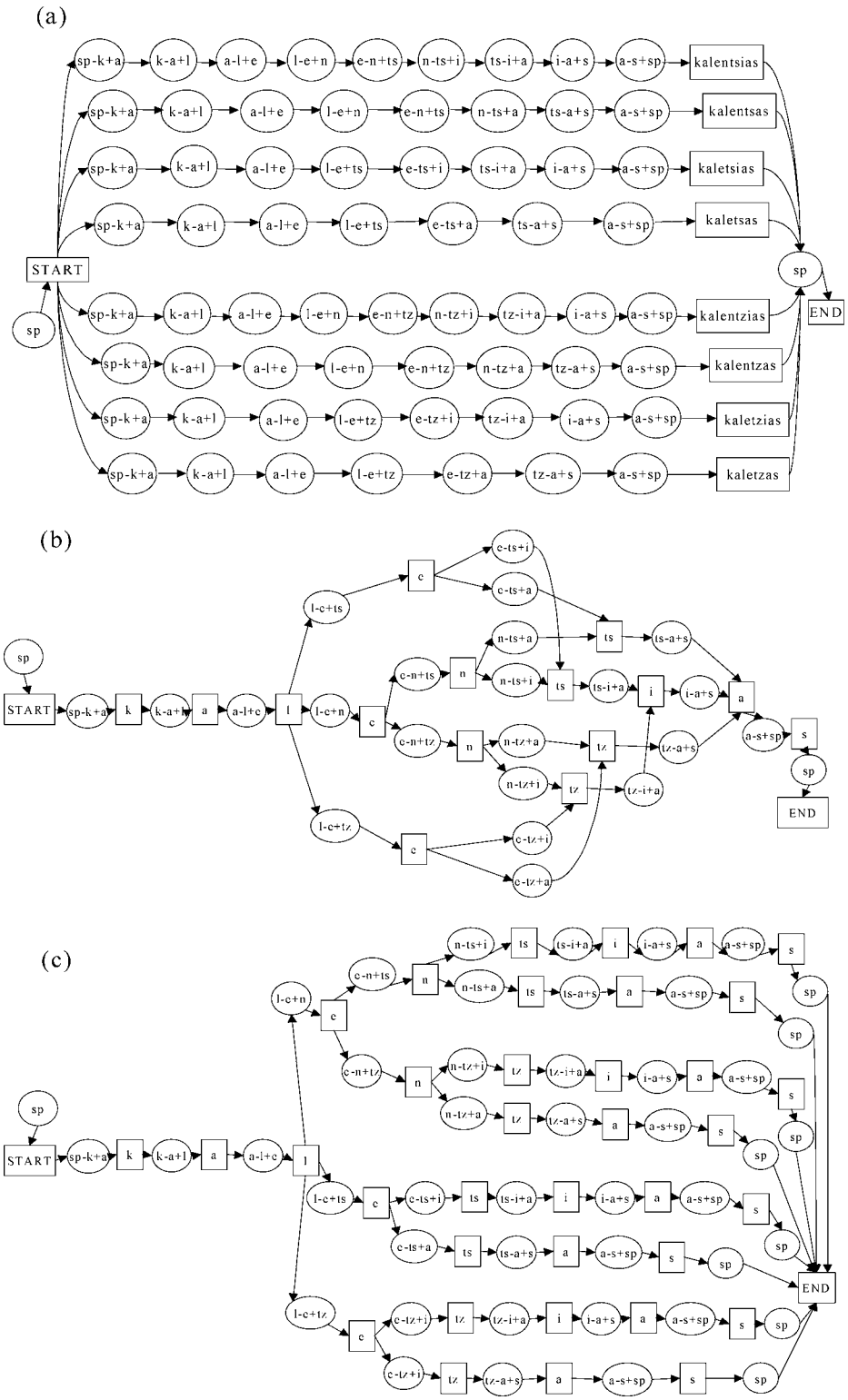

Figure 3. The expansion of (a) the full-form word network, (b) the graph-based phoneme network, and (c) the tree-based phoneme network, to triphone model nodes and word-end nodes.

The Viterbi beam search algorithm traverses the expanded network and estimates the acoustic probabilities until it reaches a word-end node. At this point, it combines the above probabilities with the language modeling probability of the word in the word-end node. In our case we do not use transition probabilities between words since each word is a monophone. Transition probabilities are only applied when the surname sub-network is connected to the rest of the language model. Thus, the final scores depend on only the acoustic probabilities and since both the word and phoneme networks give the same sequences of models in each path, the recognition accuracy is not affected. Although a phoneme network (tree or graph) becomes smaller than a full-form network only after surnames exceed a certain number, the recognition time is improved for all vocabulary sizes. This is explained, if we take into account the fact that the only reason the expanded phoneme network could have more nodes and links than the corresponding word one, is the additional number of word-end or *NULL* nodes. However, the computational cost at a word-end or *NULL* node is very small compared to the cost at a triphone model node, even if we use transition probabilities between words, which is not our case.

3.4. Processing of Phonological Rules

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 symbol to the last one. When a phoneme or a phoneme 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 *kaletsias*, which is the input string to our algorithm and we have rules

$$\#w,\ tsi\ ts,\ \#w \qquad \text{(Rule 4)}$$

$$\#w,\ ts\ tz,\ \#w \qquad \text{(Rule 5)}$$

$$\#w\text{-},\ nts\ ts,\ \#w \qquad \text{(Rule 6)}$$

$$\text{-},\ g\ k,\ \#w \qquad \text{(Rule 7)}$$

where

$$\#w = (a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z, \text{-})$$

and

$$\#w\text{-} = (a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z)$$

Rule 4 says that *tsi* can be interchanged with *ts* in any context. Rule 5 denotes that *ts* and *tz* can replace one another also in any context. Rule 6 states that *nts* can be interchanged with *ts* if the left context is not the empty string and in any right context. Finally, according to Rule 7, g can be replaced by k or vice versa when g or k are first in a word. The procedure of the rules' application is depicted in Fig. 4. The numbers in parentheses show the rule that is applied each time. The input string *kaletsias* is traversed from left to right. The first phoneme is k. The algorithm searches for a rule where k is one of the central symbols. Rules 4–6 cannot be applied but Rule 7 can. Thus we have two solutions so far:

$$g\,(\text{A1}),\ k\,(\text{A2})$$

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

$$gale\,(\text{B1}),\ kale\,(\text{B2})$$

Now that we are in t, Rule 4 is applied and we get

$$galetsi\,(\text{C1}),\ galets\,(\text{C2}),\ kaletsi\,(\text{C3}),\ kalets\,(\text{C4})$$

The pointer moves to a. We go on to find if another rule is applicable. Rule 5 is, so we get 4 additional solutions:

$$galets\,(\text{C5}),\ galetz\,(\text{C6}),\ kalets\,(\text{C7}),\ kaletz\,(\text{C8})$$

The pointer moves to i for solutions C5–C8. Again we continue for another rule that could be applied. Rule 6 is applicable resulting in the following solutions:

$$galents\,(\text{C9}),\ galets\,(\text{C10}),\ kalents\,(\text{C11}),\ kalets\,(\text{C12})$$

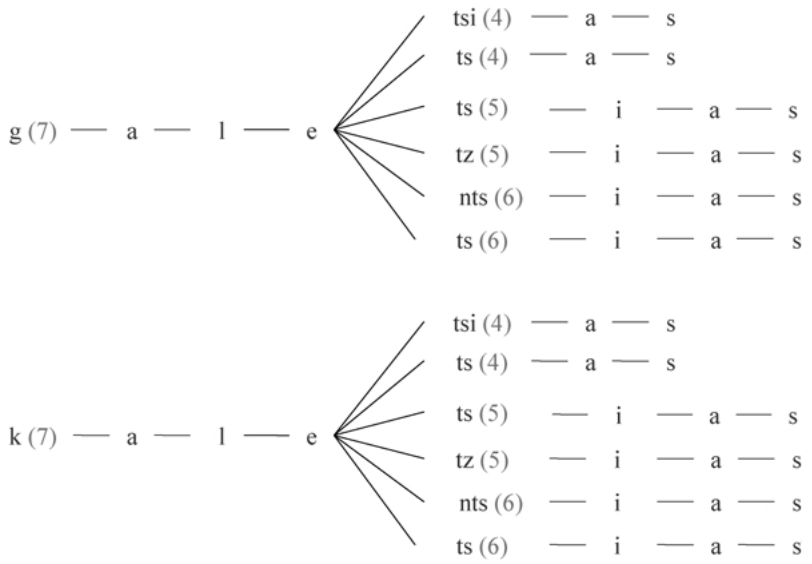

Figure 4. The application of rules for the input string *kaletsias*.

Now the pointer is at *i* for solutions C5–C12, but it was placed at *a* for C1–C4. Consequently we have to store different pointers according to the positions in the input string, where different rules are applied. In the following the algorithm processes each one of the 12 solutions we have so far. For solutions C1–C4 the pointer is at *a*, and no rule can be applied until we get to the end of the input string. Therefore, we get

galetsias (D1), *galetsas* (D2), *kaletsias* (D3), *kaletsas* (D4)

For solutions C5–C12 the pointer is at *i* and no rule can be applied until the end of the input string. Consequently, we get

galetsias (D5), *galetzias* (D6), *kaletsias* (D7), *kaletzias* (D8)
galentsias (D9), *galetsias* (D10), *kalentsias* (D11), *kaletsias* (D12)

Note that some solutions are identical, e.g., D1 D5 and D10, and D3 D7 and D12. 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 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 4 had the following form:

$$\#w, ts\,tsi, \#w \qquad \text{(Rule 4a)}$$

the algorithm would first find the sub-string *ts*. Then it would replace it with *tsi* and move the pointer to *i*. Therefore, instead of *galetsias*, *galetsas*, *kaletsias* and *kaletsas* we would get *galetsiias*, *galetsias*, *kaletsiias* and *kaletsias*. To avoid this problem, either we put in the center of the rule the sub-strings 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 sub-string 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 symbols. If no word that begins with this sub-string 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 (phoneme symbols) to decide whether a surname is valid or not.

Statistical processing of the list of most frequent surnames has also produced weights for each rule.

Suppose that we have Rules 4 and 5. We find *N1* surnames that would be similar if we interchanged *tsi* and *ts* in any context #*w*, and *N2* surnames that would be equivalent if we replaced *ts* with *tz* and vice versa 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 for which 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.

4. Experiments

4.1. Graphs vs. Full-Forms and Trees

In order to test the efficiency of graphs compared to full-forms and trees we used 106 different surnames spoken by four speakers (two male and two female). We carried out three types of experiments. In the first type we used a full-form network like the one depicted in Fig. 2(a), in the second a graph-based phoneme network (Fig. 2(c)), and in the third a tree-based phoneme network (Fig. 2(d)). Tests also differed in the number of words contained in the dictionary, that is, in the size of the full-form and phoneme networks. The performed experiments had three goals: (1) to examine how the number of nodes and links changes according to the vocabulary size and prove that after a certain point it decreases for trees and furthermore for graphs; (2) to show in practice that recognition accuracy is retained; and (3) to prove that processing time decreases for all vocabulary sizes (for trees and for graphs).

Figure 5(a) shows the number of nodes and links for nine different vocabularies described by two numbers. The first one is the number of distinct pronunciations, and the second is the number of distinct surnames. The first number is always greater than or equal to the second one because some words are pronounced in multiple ways. Figure 5(b) depicts the accuracy for different vocabulary sizes and six pruning levels. *L0* means that there is no pruning, and the search is exhaustive. As we go from *L1* to *L5* pruning increases, and more paths are abandoned before their full search. We have used two pruning thresholds, one for model nodes and the other for word-end nodes. We have chosen the two thresholds to be equal. The results of the tests confirm that the accuracy is the same for full-forms, trees and graphs in all cases, ranging from 55 correct recognitions for high pruning and large vocabularies to 103 for small or no pruning and small vocabularies.

In Fig. 6(a)–(d) we can see the absolute time (sec) that is required in average for recognizing one surname using full-forms, trees and graphs for different vocabulary sizes and pruning levels. Figure 6(d) depicts the absolute recognition time (sec) of full-forms, trees and graphs for a vocabulary of 88,000 surnames that correspond to 123,313 distinct pronunciations. The reason we have drawn a separate chart for the absolute time of this vocabulary size is that the scale of the vertical axis is different from the scale of the absolute time chart for the rest of the vocabularies. Figure 6(e)–(g) show the relative (%) recognition time gain between full-forms and trees, full-forms and graphs, and trees and graphs.

Pruning level *L3* gives the best trade-off between accuracy and processing time. According to the

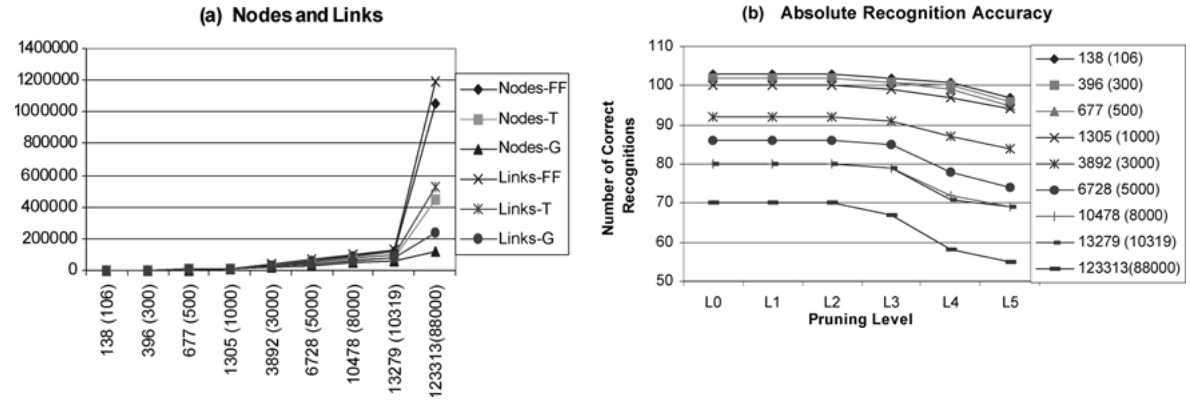

Figure 5. (a) Number of nodes and links of full-forms, trees and graphs, (b) absolute recognition accuracy of full-forms, trees and graphs.

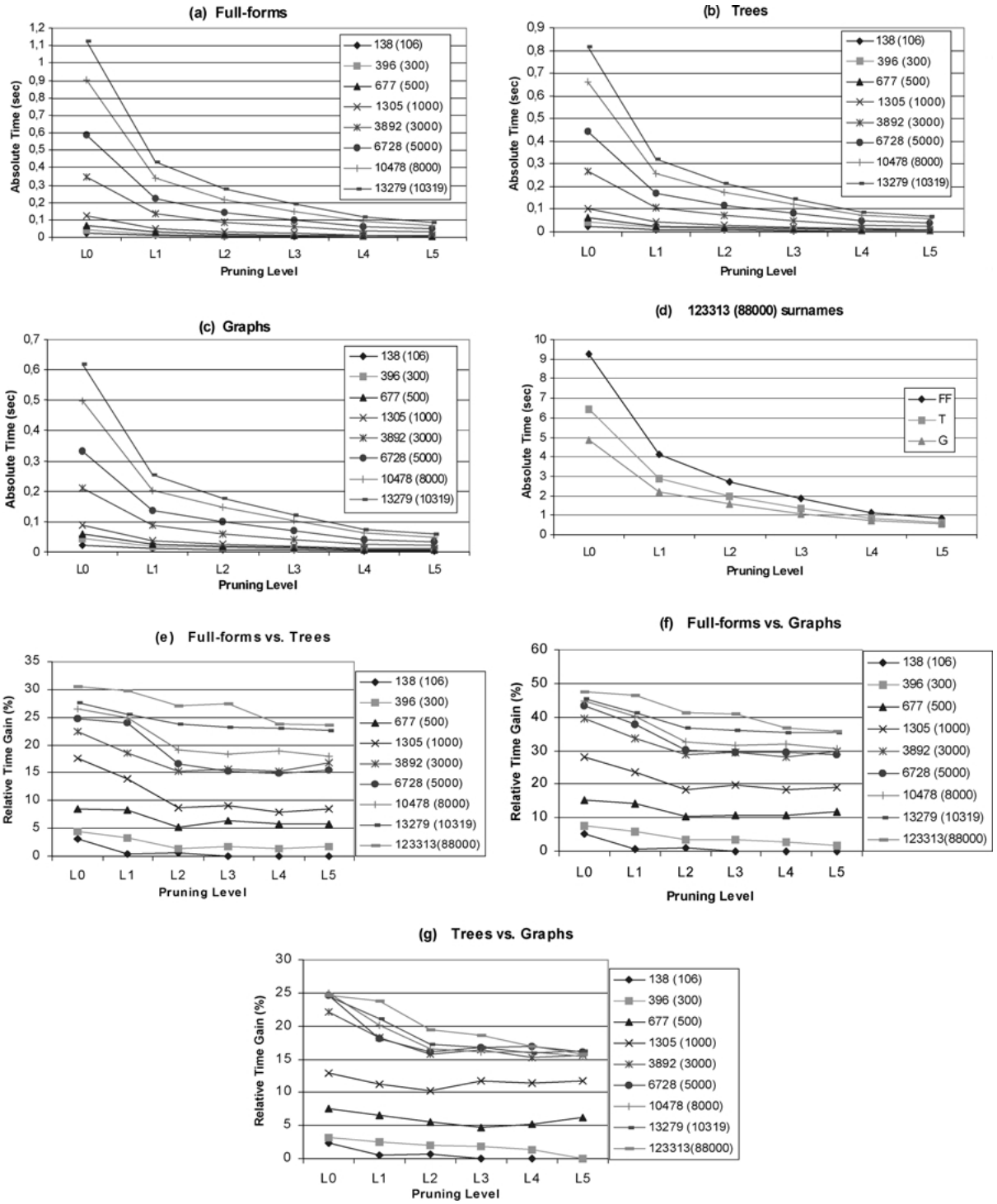

Figure 6. Absolute recognition time of full-forms (a), trees (b) and graphs (c). Absolute recognition time of full-forms, trees and graphs for the vocabulary of 123313 surnames (d). Relative recognition time gain between full-forms and trees (e), full-forms and graphs (f), trees and graphs (g).

experiments, when trees or graphs are used, if we select a word-end pruning threshold higher than the model one, accuracy drops (maximum 8% for large vocabularies). This is explained by the fact that each word is equivalent to a model, thus a greater pruning threshold for word-end nodes entails an increased pruning threshold for model nodes, even if the model pruning threshold is smaller. It should also be noted that experiments showed that if the word-end pruning threshold in full-form networks is greater than the word-end pruning in phoneme networks, while both network types have the same model pruning threshold, we get the same accuracy. In this case one would expect that since pruning increases for word networks, time would decrease. This is true, but it still remains significantly greater than the processing time of phoneme networks that have smaller pruning thresholds. Thus, even if we need smaller pruning thresholds in phoneme networks to get the same accuracy we have in word networks with higher pruning thresholds, recognition time in phoneme networks (tree-based or graph-based) is still significantly smaller. The above observation justifies the efficiency of using trees and graphs. As a general conclusion, the larger the vocabulary, the higher the absolute recognition time is, for every pruning level. Moreover, the absolute recognition times and the relative recognition time gains decrease as the pruning level gets higher for all vocabulary sizes. This is not always true for the relative time gain. The curves for the relative time gain are not always descending, which is explained by the fact that the time values have been rounded in some cases, something that can cause divergence in the final percentages, because we deal with very small time intervals.

4.2. *The Effect of Phonological Rules on Recognition Accuracy*

Field tests were carried out with 110 people to evaluate the performance of the automatic directory information system as a whole. The 76 males called the system 381 times, and the 34 females 123 times. These people were chosen to cover different ages, dialects and education levels. By that time there was also improvement in the acoustic models, which led to better recognition rates compared to the ones we had during the evaluation of full-forms, trees and graphs. The surname recognition accuracy without applying rules was 70.85%. The speech recognizer produced only the best hypothesis ($N = 1$).

Table 1. Surname recognition accuracy for different values of N (in the N-best hypotheses' list), with and without the application of phonological rules.

	Male (%)	Female (%)	Total (%)
Without phonological rules			
$N = 1$	159/69.13	98/70.00	257/69.46
$N = 3$	162/70.43	98/70.00	260/70.27
$N = 5$	163/70.87	100/71.43	263/71.08
$N = 10$	168/73.04	102/72.85	270/72.97
$N = 15$	172/74.78	104/74.28	276/74.59
$N = 20$	179/77.82	108/77.14	287/77.56
$N = 25$	186/80.86	112/80.00	298/80.54
$N = 30$	191/83.04	116/82.85	307/82.97
With phonological rules			
$N = 1$	195/84.78	119/85.00	314/84.86
$N = 3$	200/86.95	121/86.43	321/86.75
$N = 5$	202/87.82	123/87.85	325/87.83
$N = 10$	207/90.00	127/90.71	334/90.27

In order to evaluate the recognition performance after the application of phonological rules, new tests were carried out. Thus, 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, both with and without phonological rules. The results are depicted in Table 1. In each cell the first value shows the absolute number of correct recognitions and the second the corresponding percentage.

If we do not use phonological 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 3 because the computational cost became too high. When we applied phonological rules, we realized that $N = 1$ was enough to produce better results than $N = 30$ (with no phonological rules), 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 required for taking an output and applying the phonological rules. Moreover, the application of rules leads to significantly more than 30 solutions, which have the advantage of being based on language dependent data (not just the acoustic signal). Thus, the probability of including the correct surname

is higher. The results are even better when we have $N = 10$ and use phonological rules. However, in this case, as for $N = 10$ without rules, the response time is not very good. In conclusion, $N = 3$ with phonological rules is the solution that combines good recognition accuracy and real-time response. In total, there were 52 rules, which is a high number if we consider that the rules' structure allows for including many cases in the same rule by using classes. At first, we had 95 rules, but the processing time was prohibitive for real-time applications, with no gain in accuracy because most of the rules covered very rare cases. Thus, we decided to keep only the ones that covered the most frequent interchanges between phonemes and phoneme combinations.

5. Summary and Conclusions

In this paper we described two methods aiming at reducing the search space in large vocabulary speech recognition. First, we used DAWG structures in order to replace word networks with phoneme networks in such a way that all the possible paths of the phoneme networks produce the phonetic transcriptions of the words in the word networks. The DAWGs were transformed to graphs in the format expected by the decoder, where the labels were on the nodes instead of the arcs. To test the efficiency of our method we compared full-form networks, trees and graphs under the same conditions (using the same decoder and vocabularies). We proved that the size of trees and graphs is reduced after a certain point compared to full-form networks and that recognition accuracy is retained while processing time decreases significantly for all vocabulary sizes, and especially for larger ones. It was also shown that graphs are more compact than trees and lead to smaller recognition times. The aim of the second technique was to refine the N-best hypotheses' list provided by the speech recognizer by applying phonological rules. The performed experiments showed that the application of phonological rules results in better recognition accuracy compared to the cases for which no rules were 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. Both methods were applied to surname recognition in an automatic directory information system.

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

Acknowledgments

The authors would like to thank Dr. Kyriakos Sgarbas at Wire Communications Laboratory for providing the algorithm for the DAWG construction and Dr. Anastasios Tsopanoglou at Knowledge S.A. for his important contribution to this work. We would also like to thank Dr. Daryle Gardner-Bonneau and the anonymous reviewers for their valuable comments on a draft of this paper.

Note

1. EU project LE4-8315 IDAS (Interactive telephone-based Directory Assistance Services).

References

Aoe, J., Morimoto, K., and Hase, M. (1993). An algorithm for compressing common suffixes used in trie structures. *Systems and Computers in Japan*, *24*(12):31–42 (Translated from *Trans. IEICE*, *J75-D-II*(4):770–799, 1992).

Betz, M. and Hild, H. (1995). Language models for a spelled letter recognizer. *Proceedings of ICASSP*, Detroit, MI, Vol. 1, pp. 856–859.

Billi, R., Canavesio, F., and Rullent, C. (1998). Automation of Telecom Italia directory assistance service: Field trial results. *Proceedings of IVTTA*, Turin, Italy, pp. 11–16.

Chen, F.R. (1990). Identification of contextual factors for pronunciation networks. *Proceedings of ICASSP*, pp. 753–756.

Collingham, R.J., Johnson, K., Nettleton, D.J., Dempster, G., and Garigliano, R. (1997). The Durham telephone enquiry system. *International Journal of Speech Technology*, *2*(2):113–119.

Córdoba, R., San-Segundo, R., Montero, J.M., Colás, J., Ferreiros, J., Macías-Guarasa, J., and Pardo, J.M. (2001). An interactive directory assistance service for Spanish with large-vocabulary recognition. *Proceedings of Eurospeech*, Aalborg, Denmark, pp. 1279–1282.

Georgila, K., Sgarbas, K., Fakotakis, N., and Kokkinakis, G. (2000). Fast very large vocabulary recognition based on compact DAWG-structured language models. *Proceedings of ICSLP*, Beijing, China, Vol. 2, pp. 987–990.

Gopalakrisnan, P.S., Bahl, L.R., and Mercer, R.L. (1995). A tree search strategy for large vocabulary continuous speech recognition. *Proceedings of ICASSP*, Detroit, MI, Vol. 1, pp. 572–575.

Gupta, V., Robillard, S., and Pelletier, C. (1998). Automation of locality recognition in ADAS Plus. *Proceedings of IVTTA*, Turin, Italy, pp. 1–4.

Hanazawa, K., Minami, Y., and Furui, S. (1997). An efficient search method for large-vocabulary continuous-speech recognition. *Proceedings of ICASSP*, Munich, Germany, pp. 1787–1790.

Kamm, C.A., Shamieh, C.R., and Singhal, S. (1995). Speech recognition issues for directory assistance applications. *Speech Communication*, *17*:303–311.

Kaspar, B., Fries, G., Schumacher, K., and Wirth, A. (1995). FAUST—A directory-assistance demonstrator. *Proceedings of Eurospeech*, Madrid, Spain, pp. 1161–1164.

Lennig, M., Bielby, G., and Massicotte, J. (1995). Directory assistance automation in Bell Canada: Trial results. *Speech Communication*, *17*:227–234.

Mitchell, C.D. and Setlur, A.R. (1999). Improved spelling recognition using a tree-based fast lexical match. *Proceedings of ICASSP*, Phoenix, AZ.

Nguyen, L. and Schwartz, R. (1999). Single-tree method for grammar-directed search. *Proceedings of ICASSP*, Phoenix, AZ.

Phonetic Systems (2002). Searching large directories by voice. *Provided by Phonetic Systems*.

Ramabhadran, B., Bahl, L.R., deSouza, P.V., and Padmanabhan, M. (1998). Acoustics-only based automatic phonetic baseform generation. *Proceedings of ICASSP*, Seatlle, WA, Vol. 1, pp. 309–312.

Schmid, P., Cole, R., and Fanty, M. (1993). Automatically generated word pronunciations from phoneme classifier output. *Proceedings of ICASSP*, Minneapolis, MN, Vol. 2, pp. 223–226.

Schramm, H., Rueber, B., and Kellner, A. (2000). Strategies for name recognition in automatic directory assistance systems. *Speech Communication*, *31*:329–338.

Seide, F. and Kellner, A. (1997). Towards an automated directory information system. *Proceedings of Eurospeech*, Rhodes, Greece, Vol. 3, pp. 1327–1330.

Sgarbas, K., Fakotakis, N., and Kokkinakis, G. (1995). Two algorithms for incremental construction of directed acyclic word graphs. *International Journal on Artificial Intelligence Tools*, *4*(3): 369–381.

Sgarbas, K., Fakotakis, N., and Kokkinakis, G. (2001). Incremental construction of compact acyclic NFAs. *Proceedings of ACL-EACL*, Toulouse, France, pp. 482–489.

Suontausta, J., Häkkinen, J., and Viikki, O. (2000). Fast decoding in large vocabulary name dialing. *Proceedings of ICASSP*, Istanbul, Turkey, 2000.

Van den Heuvel, H., Moreno, A., Omologo, M., Richard G., and Sanders, E. (2001). Annotation in the SpeechDat projects. *International Journal of Speech Technology*, *4*(2):127–143.

Whittaker, S.J. and Attwater, D.J. (1995). Advanced speech applications—The integration of speech technology into complex services. *ESCA Workshop on Spoken Dialogue Systems—Theory and Application*, Visgø, Denmark, pp. 113–116.

Young, S., Odell, J., Ollason, D., Valtchev, V., and Woodland, P. (1997). *The HTK Book* (user manual), Entropic Cambridge Research Laboratory, Cambridge.