Speaker Classification for Mobile Devices

Michael Feld, Christian Müller
German Research Center for Artificial Intelligence (DFKI)
Saarbrücken, Germany
{michael.feld,christian.mueller}@dfki.de

Abstract

User adaptivity is a key topic in the context of mobile devices and applications, and speech is one the sources of information which has more recently been discovered for this purpose. While considerable work has already been done in both finding algorithms and designing well-performing implementations for this speaker classification task on the desktop platform as part of the AGENDER approach, efforts to bring the results to portable platforms in a working framework have been rather scarce so far. This work seeks to state the major aspects that make mobile speaker classification different from its desktop counterpart, and proposes a number of changes and enhancements to the existing infrastructure to fulfill the requirements emerging from it.

1 Introduction

Our work is based on the AGENDER [6] speaker classification approach. In AGENDER, a speech sample is classified according to several characteristics, originally being age and gender (hence the name), but now extended to language, noise context, and to even further aspects in the future. It has been found an efficient way to support user modeling in situations where no or little explicit information about the user is available, and quite a number of these scenarios involve mobile devices. One example would be a navigation application running on a device which is owned by a third party and only handed to the user temporarily in order to complete the navigation task. Another example is a shopping scenario where the user utilizes a handheld device to access services and retrieve additional information about products in a physical store [4]. In the last case, the application is created by the store owner and thus we cannot assume prior knowledge about the user.

Analyses of large corpora of labeled speakers such as *SpeechDat* have revealed that there are indeed speech features like pitch, jitter and shimmer, which convey sufficient information to discriminate between eight classes (four age groups and two genders) with a promising accuracy of 63.5% (see [8] for the complete results). Similarly, on *GlobalPhone*, the three languages German, Turkish, and English could be classified using an n-gram approach with an accuracy of 76.7%. Feature extraction was performed with the tool *Praat*[1] [1] for prosodic features and *Sphinx 2* [5] for phonemes, while classification was done with several algorithms from the *WEKA* Machine Learning library [9].

In a mobile scenario, there are special requirements and technical constraints that a concept like AGENDER has to take into account. This work summarizes the major issues and presents a framework that enables device-based speaker classification.

2 The Agender Approach

A person's recorded voice contains a lot of information about the speaker. This includes sociological properties such as language and accent, mental properties like cognitive load and emotion, physiological properties such as height and consumed substances, and others.

In a first step to determine the phonological attributes of these features in speech, data collection of several large corpora of labeled speakers and their empirical analysis have been performed [6, 7]. Through these studies, several prosodic features such as *pitch*, *jitter*, *intensity*, *shimmer* and *harmonics-to-noise ratio*, have been found which convey sufficient information to distinguish between genders and ages. For age, the current version uses four classes: *Children* (up to 12 years), *Teenagers* (13-20 years), *young Adults* (21-64 years) and *Seniors* (65 years and older). The choice of these boundaries can be primarily attributed to the biological changes that typically occur to the human anatomy around these ages, especially the vocal tract, and that affect the characteristics of speech, and thus increase the classification accuracy. They are also reasonable classes for many application scenarios. However, given that sufficient training material is available, other class separations

[1] `http://www.praat.org`

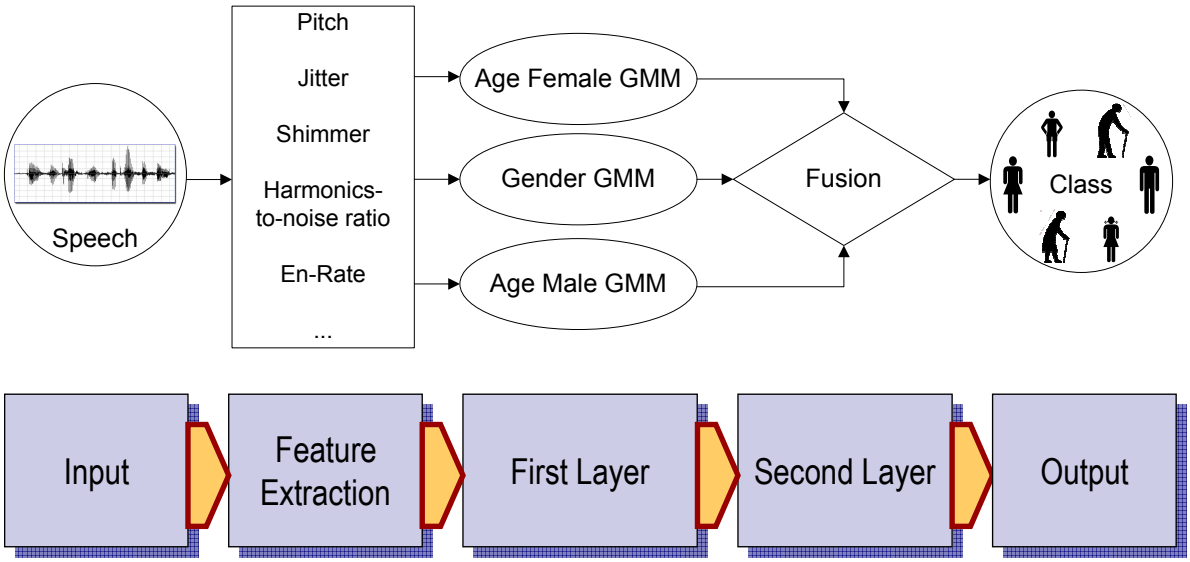

Figure 1. A sample pattern classification system implementing AGENDER. There is one classifier for gender and two gender-specific classifiers for age. Their results are combined on the second layer.

may also be used. Another result of the data analysis was the understanding that voices do not only differ between the genders and between different ages, but that also a gender-specific vocal aging can be witnessed, a fact which can be exploited by combining several classifiers.

Using methods from signal processing implemented in the tool *Praat*, common statistics based on these features (e.g. mean and standard deviation) were extracted on the available corpus data. The resulting data was then used to train models for each of the speaker classes, which consist of an *age* and a *gender* component (e.g. AF = Adult Female). Several well-known machine learning methods have been investigated, in particular Naive Bayes, Gaussian Mixture Models (GMM), k-Nearest-Neighbor, C 4.5 Decision Trees, Support Vector Machines and Artificial Neural Networks (ANN). The highest accuracy was in most tests reached by the ANN classifier (see [8] p. 121), but other factors like speed and memory consumption may also affect the choice, especially in mobile scenarios. In subsequent tests, it also became apparent that some classes, including those spanning different speaker properties, could be grouped to form a single combined class that resulted in a better overall performance for a specific feature set. For example, one classifier discriminates between the three classes *children*, *female adults* and *male adults or seniors*.

The results from multiple classifiers extracted on a "first layer" can be combined on a "second layer" using a Dynamic Bayesian Network [2]. This method can also be used to exploit the aforementioned fact of gender-dependant vocal aging by modeling the probability of a gender-specific age classifier as dependant on the probability output of the *gender* classifier. This improves performance because currently, gender can be classified with a much higher accuracy than age on unfiltered data. Additionally, the aspect of time is incorporated into the network when multiple utterances of the same speaker are considered, so that the final probability should converge and reduce classification errors.

Figure 1 shows an example classification setup consisting of the phases feature extraction, classification ("first layer"), and post-processing ("second layer"). It distinguishes six age/gender classes and employs three classifiers.

3 Requirements for the Mobile Scenario

There are quite a few differences between the various types of scenarios in which AGENDER can be used, e.g. the mobile scenario, a non-portable embedded scenario (such as public terminals), and large-scale server-based applications. Hence, adapting the existing AGENDER platform to support a portable experience while preserving its major characteristics like accuracy and runtime did pose a number of challenges to both the architecture as well as the engineering. As part of this work, the following main requirements for a mobile speaker classification framework have been identified and investigated.

First, a compact implementation for the classification process that will fit the mobile application concept is needed. On the desktop, the classification process involved calling several sub-processes, running them in parallel, each of them creating many temporary files, and collecting the

results afterwards. This behavior and the fact that the first implementation provided only Java-based access were found to be not fulfilling this requirement in our analysis. In this phase, we came up with the concept of *Embedded Modules*, which is described in other recent work [3] and which represents a generic and portable approach to embedded machine learning scenarios. We will elaborate on how this concept aids us in our task in the next section.

A second requirement is that our mobile classifiers need to cope with platform and device limitations. There are several conditions applying specifically to portable devices that need to be taken into account. One such factor is memory, which is usually much less on mobile devices than on desktop PCs. Also, we can expect the CPU to be far less powerful, which in the worst case could mean an intolerable increase in classification time. Then, because the user is expected to be constantly changing his or her location, we cannot assume a permanent network connection in the server-based classification scenario, but need to be able to handle sporadic connections. There are of course other hardware properties unique to mobile platforms, such as the size factor, but the majority of those which have not already been mentioned lie in the responsibility of the application.

In stand-alone, local scenarios, classification should be able to run fully on the device. This corresponds largely to the desktop-based classification, except that all processing is done on a machine with much less computational power. However, if there is a connection to a more powerful server for classification at our disposal, we also want to be able to take advantage of this additional resource. Hence, as a further requirement, our architecture should support both variants, i.e. client-based and sever-based classification.

The last aspect relates to the tools which are used to build the classifiers. The AGENDER approach that exists today is implemented as part of an integrated development environment called AGENDERIDE, which supports the developer with training, evaluation, and building of classification modules. While not specifically a requirement of mobile platforms, the addition of a new platform, especially with such a discrepancy in characteristics like the mobile platform, involves changes in the way code is generated and compiled. Because we not working with a static application but dynamically built modules, this cannot be simply considered a standard porting problem. Our task is to extend this architecture to also build modules that can run on mobile platforms such as PocketPC and Smartphone while maintaining compatibility with the version for personal computers and servers.

4 Design Concepts

The overall architecture which we created to comply with the requirements outlined in the previous section is depicted in Figure 2. The development environment AGENDERIDE is the basis for creating classification modules that will run on mobile devices. We use the existing tools for corpus management, feature extraction, classifier training and evaluation to compose the modules which are going to solve the respective classification problems.

An Embedded Module is made up from various components such as classifiers and pre-processors, which are connected together and equipped with an external interface. The module core takes care of i/o management, component caching, tracing and parallelization. It can be integrated into applications through means of static C/C++ linking, dynamic linking (DLLs) and common language interface (Java, .NET). This flexible design is especially helpful for mobile applications. For example, certain devices may require specific pre-processing layers to reduce acoustic effects introduced by their microphones, while others do not. Also, for learning methods that are implemented in hardware, individual layers of the classification process can be replaced without affecting the other layers and without having to redesign the architecture.

The Embedded Module is built using AGENDERIDE's build engine. This module build engine is extended to support multiple platforms and multi-targeting, such that the same module configuration can be compiled for different platforms in one step. As each module is compiled individually, any optimizations or configuration options for specific platforms will be compiled into the object code, resulting in less and faster code. The embedded module source code frame is ported to the Windows CE platform, which we used to obtain our results.

Initial tests with a classifier built for the PocketPC showed that even for short utterances, the module required a considerable amount of time (usually more than a minute) to complete. Timing analyses have been performed and some critical routines have been optimized to require less computational power at the cost of some accuracy. The most notable gain was achieved by reducing interpolation in the *Praat* feature extraction routines for all pitch-based features. In order to reduce problems resulting from higher classification times on mobile devices, we have also provided the ability for applications to do background user model adaptation in cases whether this does not negatively impact the user experience, i.e. classify a speech sample in the background and adapt the application once the user profile is updated.

To comply with the reduced memory of the mobile device, several options that control caching have been added. For example, it can be configured by the module designer which classifier models are kept in memory and which are loaded on access. Also, clean-up tasks can be executed more frequently.

Our approach intrinsically supports the local classifica-

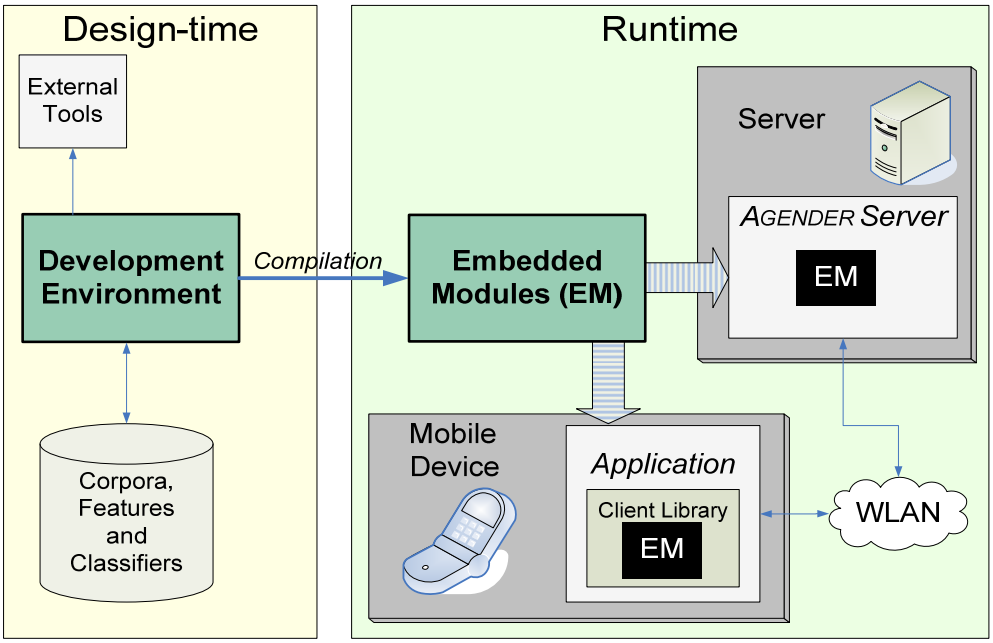

Figure 2. Architecture for device-based classification.

tion scenario. To add the option of server-based classification as described earlier, we included an AGENDER classification server on the one hand and a wrapper client library that can connect to that server on the other hand. Both are featuring the same module configuration built for different platforms and with different optimizations. Server-based classification is transparent to the application, i.e. there is no difference in calls depending on whether local or remote classification is active. The library can even switch between both on-the-fly, which mitigates the effects of connection loss (e.g. when connected over a wireless LAN) since the application will continue classifying locally. The connection is established through TCP/IP sockets and uses a compact custom protocol.

5 Results

Using this new architecture, some preliminary experiments to measure the classification time have been performed using four different PocketPC devices (see Table 1). The current build engine supports the platforms *Win32* (X86 processors), *Windows CE 3.0* and *Windows Mobile 5.0* (both PocketPC and Smartphone). In this evaluation, utterances of different lengths and audio formats have been classified using a GMM very close to the one in Figure 1 on each of the devices, while the time needed for the complete classification pipeline (e.g. including feature extraction and post-processing) was measured.

Figure 3 illustrates the results. In two of the charts, we are comparing the numbers stemming from the optimized mobile versions of the components with the numbers obtained using the desktop version of the classifiers with only the required porting code applied. Our experiments also revealed that feature extraction accounts for more than 95% of the processing time for this type of classifier.

It is clear that more evaluation is needed in order to find a tendency in how fast processing time approaches real time for newer devices, and to determine the areas where optimizations of the mobile versions of the modules are most effective.

6 Summary and Future Work

In this paper, we have shown that speaker classification is indeed possible on mobile platforms. Moreover, the approach presented here is advantageous when it comes to compatibility with desktop applications and integration of the modules. The two suggested scenarios are client-only and server-based with a client fallback option.

The device hardware is currently the limiting factor of the implementation, especially CPU power. We have seen that without any optimizations, classification times are too high for most applications. With the optimizations enabled, they are almost at real-time on newer devices for telephone-quality audio. In this context, the audio quality aspect should not be given too much priority because very often, the recording hardware on mobile devices does not provide high fidelity sound anyway. With older and slower devices,

Device name	Processor	CPU frequency	Platform
HP Jornada 568	ARM SA1110	206 MHz	Windows CE 3.0
HP Ipaq H6300	TI OMAP 1510	200 MHz	Windows CE 4.2
HP Ipaq H5450	Intel PXA250	400 MHz	Windows CE 4.2
HP Ipaq HX4700	Intel PXA270	624 MHz	Windows CE 4.21

Table 1. Mobile devices used in our evaluation.

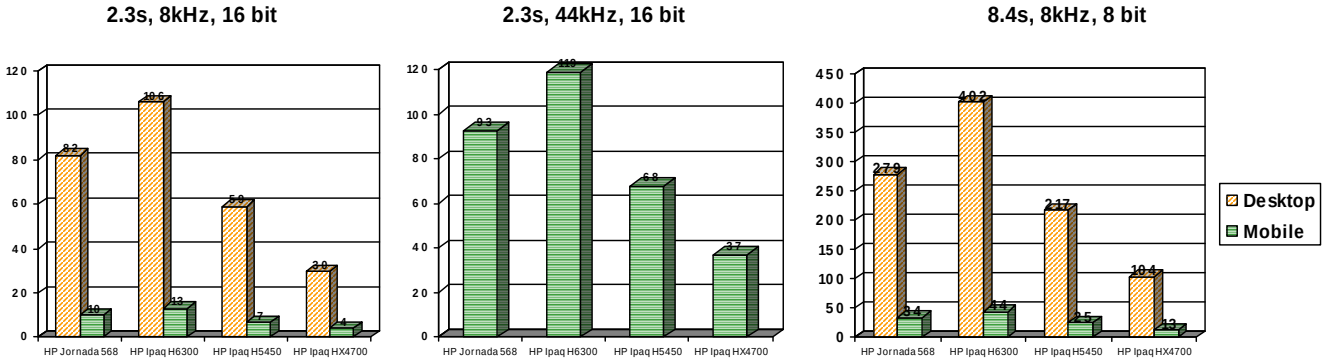

Figure 3. Benchmark results for different PocketPCs. Each bar displays the time in seconds needed to process an utterance.

scenarios for background adaptation are still possible.

In future work, one important aspect will be to improve the performance of device-based classification. Although newer devices will also provide more powerful CPUs, we think that optimizing the algorithms can also help with our goal to drive classification times down to a fraction of real-time.

Additionally, capabilities and hardware configuration of devices should be exploited even more. For example, device families with different types of microphones should be able to choose between different classifiers to match the recording quality of the hardware.

References

[1] P. Boersma. PRAAT, a system for doing phonetics by computer. *Glot International*, 9(5):341–345, 2001.

[2] B. Brandherm. *Eingebettete dynamische Bayessche Netze nter Ordnung [Embedded dynamic Bayesian networks of n-th order]*. PhD thesis, Computer Science Institute, University of the Saarland, Germany, 2006.

[3] M. Feld. Embedded Modules for Speaker Classification. In *Proceedings of the IEEE International Conference on Semantic Computing 2008 (ICSC 2008)*, Santa Clara, CA, USA, August 2008. (to appear).

[4] M. Feld and G. Kahl. Integrated Speaker Classification for Mobile Shopping Applications. In *Proceedings of the 5th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH 2008)*, Hannover, Germany, July 2008. (to appear).

[5] X. Huang, F. Alleva, H.-W. Hon, M.-Y. Hwang, and R. Rosenfeld. The SPHINX-II speech recognition system: an overview. *Computer Speech and Language*, 7(2):137–148, 1993.

[6] C. Müller. *Zweistufige kontextsensitive Sprecherklassifikation am Beispiel von Alter und Geschlecht [Two-layered Context-Sensitive Speaker Classification on the Example of Age and Gender]*. PhD thesis, Computer Science Institute, University of the Saarland, Germany, 2005.

[7] C. Müller. Automatic Recognition of Speakers Age and Gender on the Basis of Empirical Studies. In *Proceedings of the 9th International Conference on Spoken Language Processing (Interspeech 2006 – ICSLP)*, Pittsburg, PA, USA, 2006.

[8] C. Müller and M. Feld. Towards a Multilingual Approach on Speaker Classification. In *Proceedings of the 11th International Conference "Speech and Computer" SPECOM 2006*, pages 120 – 124, St. Petersburg, Russia, 2006. Anatolya Publishers.

[9] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2 edition, 2005.