Experimental Results and Statistics in the Implementation of the Modular Audio Recognition Framework’s API for Text-Independent Speaker Identification

Serguei A. Mokhov
1455 De Maisonneuve Blvd. W.
Department of Computer Science and Software Engineering
Concordia University, Montreal, Quebec, Canada, H3G 1M8
mokhov@cse.concordia.ca

ABSTRACT
In this paper we present some experimental audio signal processing results for speaker identification by using the API of the Modular Audio Recognition Framework (MARF), which as an open-source research platform implemented in Java. MARF is used to evaluate various pattern-recognition algorithms and beyond within and outside the pipeline.

Keywords: text-independent speaker identification, Modular Audio Recognition Framework (MARF), signal processing, frameworks, pattern recognition, algorithms, API, audio processing

1. INTRODUCTION
This paper focuses on the experimental results of using the Modular Audio Recognition Framework (MARF) and its speaker identification application [20] implemented in Java. This is a targeted elaboration and refinement from the previous related works on MARF [10, 11] and others. MARF is an open-source project hosted at sourceforge.net [23] with 29,000+ downloads to date1.

MARF Overview
MARF constitutes a collection of general-purpose pattern recognition APIs and their concrete realization. At its origins, MARF was developed to showcase the workings of the implemented algorithms, and, at the same time, provide researchers with a platform to test existing and new algorithms against each other in various performance metrics (e.g. run-time and memory consumption) using a common homogeneous platform. MARF, its sub-frameworks, and applications originally evolved around the concept of audio recognition, but are not restricted to that topic due to their generality as well as that of the implemented algorithms. As of this writing, MARF covers a pattern-recognition pipeline (loading, preprocessing, feature extraction, training and classification) as shown in Figure 1, various algorithms, PureData [22] plugin [12], natural-language processing, and distributed versions [9, 17, 15].

MARF Design Overview

Pipeline: The conceptual pattern recognition pipeline shown in Figure 12 depicts the basic data flow between the stages. The inner boxes represent most of the available concrete module implementations or stubs. The grayed-out boxes are either the stubs or partly implemented. The white boxes signify implemented algorithms. Generally, the whole pattern recognition process starts by loading a sample (e.g. an audio recording in a waveform), preprocessing it, then extracting most prominent features, and finally either training the system such that the system either learns a new set of features of a given subject or actually classify and identify what/who the subject is. The outcome of training is either a collection of some form of feature vectors or their mean or median clusters, which are stored per subject learned. The outcome of classification is a 32-bit unique integer usually indicating who/what the subject the system believes is. Some of the details of such processing of classification are illustrated on the actual sequence of events and method calls within the main MARF module is shown in Figure 22.

Applications: There is a vast number of possible applications that can be written based on MARF in its current form. Within the MARF’s CVS repository itself [23], there are four major and many minor test and demo applications that are bundled with MARF upon a release (or released independently sometimes). The four major applications include the Text-Independent Speaker Identification, Language Identification, Probabilistic Parsing, and Zipf’s Law analysis. The former is the most developed application that is used in a variety of tests of the MARF’s features, including the exhaustive testing of the implemented algorithms with the main goal to illustrate the best available configuration of the preprocessing, feature extraction, and classification algorithms that yields the best accuracy thereby helping the researchers in the field to focus on the algorithms they need for their particular problem in their domain [11]. In Figure 3 is the high-level UML overview of the interfaces and abstract modules of MARF [23], their interaction, and the use by the applications. Further in this work, we are focusing on the first of the listed applications, namely the speaker identification application SpeakerIdentApp [20].

Algorithms
On top of the framework’s API itself, MARF has several actual implementations of a number of algorithms to demonstrate its abilities in various pipeline stages and supporting modules. The below is the incomplete list of implemented algorithms corresponding to the Figure 1 with a very brief description:
• Linear predictive coding (LPC) – used in feature extraction.
• The Fast Fourier transform (FFT) [2] – used in the FFT-based filtering as well as feature extraction.
• A number of distance-based classifiers (Euclidean, Chebyshev, Minkowski [1], Mahalanobis [8], Diff (internally developed within the project, roughly similar in behavior to the UNIX/Linux diff utility [7]), and Hamming [4]).
• Cosine similarity measure [3], which was explored in depth in [6] and contributes to the best accuracy in this work in many configurations.
• Artificial neural network – used in classification.
• Zipf’s Law-based classifier [25].
• General probability classifier.
• Continuous Fraction Expansion (CFE)-based filters [5].
• A number of math-related utilities, e.g. for matrix and vector processing, including complex numbers matrix and vector operations, and statistical estimators used in smoothing of sparse matrices (e.g. in probabilistic matrices or Mahalanobis distance’s covariance matrix), and many others [11].

2. EXPERIMENTS AND RESULTS

The SpeakerIdentApp application collects a variety of statistics on the number of successful guesses as well as a second best approach. It also makes the measurements of a run-time of each configuration. The testing is typically done exhaustively through a script for all possible available configurations while analyzing wave forms of the voice sample set. In the experiments in [10, 11] it was found that the second best approach is an interesting result: if we don’t “guess” correctly from the first time, likely the next guess in line (ordered by likelihood based on the particular classifier category or type) is correct. There is a parallel of this notion in humans: an adult son and the father or an adult daughter and the mother, or several similar-age brothers or sisters are known to be mistaken for one another sometimes when listened to on the phone by another person, yielding the second guess is usually correct. Therefore, we provide this measure of interest alongside the first guess. The count of correct guesses is augmented each time the speaker identified matched up with the speaker expected (the expected ID is known for the purpose validating results from the speakers database we are using). In Figure 4 is a sample classification run of SpeakerIdentApp highlighting the discussed points.

---

File: testing-samples/serge-label.wav
Config: -norm -fft -cos
Processing time: 0d:0h:0m:0s:78ms:78ms
Speaker’s ID: 1
Speaker identified: Serguei Mokhov
Expected Speaker’s ID: 1
Expected Speaker: Serguei Mokhov
Second Best ID: 8
Second Best Name: Alexandr Mokhov
Date/time: Thu Feb 14 05:26:29 EST 2008
---

Figure 4: Sample output of single classification.

In Table 1 and Table 2 are excerpts of the top results for the first and second best outcomes produced after the system models were trained on 319 voice samples of 28 speakers and tested on the “unseen” by the system 32 testing samples. Each option constitutes a parameter to the application that selects an appropriate algorithm at run-time. The summary of all options is provided further.

Application Options: To give the reader some clue of what the options mean when reviewing the resulting tables, we briefly summarize their meaning here: -norm states to normalize the data; -raw is a preprocessing “no-op” that just passes the data through without any actual preprocessing; -silence indicates to remove the silence gaps; -endp means to use endpointing; -low means...
Figure 2: MARF’s Pattern Recognition Pipeline Sequence Diagram
to use the low-pass FFT filter (similarly -band, -bandstop, -bandpass, -high mean the remaining types of the FFT filters); -noise tells to remove the noise currently implemented as the low-pass FFT filter (thus, the -low and -noise options at the current implementation are somewhat equivalent); -fft or -lpc mean to use FFT or LPC for feature extraction; -aggr aggregates the features collected by FFT and LPC concatenated in one feature vector; -cos means the use of the cosine similarity measure for classification; -eucl, -diff, -cheb, and -mink indicate to use the Euclidean, Diff, Chebyshev, and Minkowski distance classifiers respectively. There are more options to the application than this, but the complete list is trimmed to conserve space and also to only list the options that appear in the results. For the complete set, please refer to the application’s manual within MARF’s documentation [23] or the application itself [20].

Module Parameters: All modules were executed at their default parameters (which can also be varied by the application when desired, but due to the large amount of such variations we provide no statistics on this). Please refer to the project documentation on the available parameters [23]. What follows is a very short summary of the default parameters used for some concrete modules:

- WAVLoader – the default quality of the WAV files used in the experiment is are 8000 Hz, mono, 2 bytes per sample, PCM-encoded. This sampling rate resembles more phone conversations where one domain of applications of this work is used.
- LPC – has 20 poles (and therefore 20 features)
- FFT – does 512-feature FFT analysis
- MinkowskiDistance – has a default of Minkowski factor $r = 4$
- FeatureExtractionAggregator – concatenates the default processing of FFT and LPC (hence 522 features)

3. CONCLUSION

Analysis: From our experiments it is clear that most accurate configurations involved the cosine similarity measure classifier. It is also shown that the silence removal generally helps to extract more distinct features of individual speakers (people are all silent in a similar if not the same way deferring the differences to the background noise), and 512 features of FFT wins the race by capturing the largest spectrum of voice frequencies ideal for cosine similarity. The band-stop FFT filter, as opposed to just the low-pass filter contributed to better accuracy by keeping the low and high frequencies of the spectrum, that, depending on the gender and the pitch, provide more discriminative power. Thus, this type of algorithms is better suited when the signature of the speaker is the typical frequencies of their voice. The follow-up on this work that used to be a part of the future work before but has come to pass by the time this work is published, are gender and spoken accent identification using mean and median feature clusters [13, 14] compared to that of the speaker identification. It should also be noted the resulting tables do not cover fully all the implemented algorithms. The primary reason is that they are quite slow sometimes especially when dealing with large number of features resulting in large matrices and the complex operations on them or large fully-interconnected neural networks. The configurations with algorithms not included in the present statistics are:

- CFE filters (band-pass, low-pass, high-pass, band-stop)
- Neural Network
- Mahalanobis distance

Not Just Audio: The word “audio” in the framework’s name and abbreviation may make the reader to believe that the framework is only constrained to the audio signal processing, which is not ex-
The document appears to be a collection of configurations, possibly for a machine learning or data processing task, given the technical terms and numbers. The text seems to be a table, but it is not clearly structured. Without clearer formatting, it's challenging to accurately parse and transcribe. It contains terms such as `aggr`, `eucl`, `fft`, `lpc`, `cheb`, `norm`, `cos`, `eucl`, `fft`, `cheb`, `norm`, `cos`, `eucl`, and includes phrases like `run #`, `silence`, `raw`, `aggr`, and `eucl`. It also mentions `Table 1: Top 100 Most Accurate Configurations` and `Table 2: Top 100 “Second Guess” Configurations`. The numbers and technical terminology suggest it's related to data processing or machine learning, but without clearer structure, it's difficult to provide a detailed transcription.
Acknowledgments: ready present, some of the work can be really as minimal as just some open-source tools with compatible licenses (BSD) are available, for speech-to-text, GUI, and the MARFL intensional scripting language. Currently, the project is further streamlining it, and implementation of the missing enhancements and feature requests within the project; the future Future Work: While there is a vast TODO wish list of desired enhancements and feature requests within the project, the future work will largely focus on improving the quality of the existing code, further streamlining it, and implementation of the missing essential features. Speech-related aspects along the MARF applications, a speech recognition and generation (text-to-speech and speech-to-text), GUI, and the MARFL intensional scripting language (to create MARF applications through a script) will be a major focus in the following versions. Given that there are some open-source tools with compatible licenses (BSD) are already present, some of the work can be really as minimal as just writing a few plug-in wrappers for packages like CMU Sphinx that already provide a powerful speech-to-text facility.

Future Work: While there is a vast TODO wish list of desired enhancements and feature requests within the project, the future work will largely focus on improving the quality of the existing code, further streamlining it, and implementation of the missing essential features. Speech-related aspects along the MARF applications, a speech recognition and generation (text-to-speech and speech-to-text), GUI, and the MARFL intensional scripting language (to create MARF applications through a script) will be a major focus in the following versions. Given that there are some open-source tools with compatible licenses (BSD) are already present, some of the work can be really as minimal as just writing a few plug-in wrappers for packages like CMU Sphinx that already provide a powerful speech-to-text facility.

Acknowledgments: Here we would like to acknowledge the contributions from the three original MARF founders, namely, Stephen Sinclair, Ian Clement, Dimitrios Nicolacopoulos as well as all subsequent contributors to the project as well as speakers who donated their voice samples. We would like to thank SourceForge.net and its maintainers for providing hosting and essential services for the open-source community. This research and development work was funded in part by the Faculty of Engineering and Computer Science of Concordia University, Montreal, Canada. We would also like to acknowledge the reviewers who took time to review this work.

4. REFERENCES