The Berkeley Phonetics Machine

Ronald L. Sprouse, Keith Johnson

UC Berkeley

ronald@berkeley.edu, keithjohnson@berkeley.edu

Abstract

The Berkeley Phonetics Machine is a Linux virtual machine image produced and used by the UC Berkeley Phonology Lab as a platform for phonetic research. It contains a full data analysis stack based on Python and R and also specialized tools for phonetic research. The machine is designed as a flexible and productive platform for established and novel research agendas that can be easily shared and reproduced. We list the software available in the machine, which includes many command-line tools for acoustic analysis and media file manipulation, as well as specialized Python libraries. We also discuss the use of this machine in the Phonology Lab and in phonetics courses. The overall experience with the machine has been positive, as faculty and graduate students are able to share and execute scripts in a common working environment. Undergraduate students have less opportunity to master the virtual machine environment but benefit from simplified instructions and fewer installation and operating problems. The primary difficulty that we have encountered has been with a few underpowered student computers that cannot run the virutual machine or do not run it well.
Index Terms: phonetics, phonetic research, virtual machine

1. Introduction

The Berkeley Phonetics Machine (BPM) is an Ubuntu Linux [1] virtual machine image for VirtualBox [2], produced and used by the UC Berkeley Phonology Lab (Phonlab) as a platform for phonetic research. It contains a full data analysis stack in the general-purpose scripting and statistical packages Python and R, which it shares with the related Berkeley Common Environment (BCE) used in various instructional and research environments on the UC Berkeley campus. The BPM additions to this stack comprise software packages for phonetic analysis and general manipulation of audio and video media files.

While the BPM is specifically designed to meet the needs of Phonlab users and to take advantage of the local computing and instructional environment, we believe the tools included in the BPM can meet the needs of speech researchers outside of UC Berkeley, and the machine image is available to the public.

2. Design goals

The BPM is designed to encourage *shareable*, *reproducible* research using *productive* workflows. We think of the BPM as establishing a common vocabulary that researchers can employ; by focusing on a standard set of analytic tools that can be combined in novel ways, the BPM accommodates a wide variety of research agendas.

The BPM environment emphasizes scripting for most data analysis tasks. Graphical tools are useful for data exploration, but scripts have several advantages for large scale analyses—by their nature they explicitly record the steps leading to a result; they can automate the analysis of large speech corpora [3]; and they are easily shared with other researchers, who can repeat an experiment or adapt them to new datasets.

These goals are reflected in the specific choices made in developing the BPM:

- Python as the default scripting language.
- Whenever possible provide royalty-free software that is not encumbered by patents or non-free license restrictions.
- Emphasis on command-line tools that can be controlled in Python scripts.
- Preference for small utilities that do one task well.
- Compatibility with BCE to take advantage of research and instructional opportunities on the UC Berkeley campus not specifically related to speech research.

The idea behind the BPM originated during an undergraduate course in phonetics in spring 2014. At that time it was an independent effort, and shortly after that course we decided to coordinate our efforts with BCE, which was a related effort starting up more-or-less contemporaneously. Since the first release of the BPM in spring 2015 it has been used in two more phonetics courses, and it is a standard operating environment of almost all Phonlab researchers.

3. Available software

The BPM consists of a data analysis stack provided by the BCE project, plus BPM additions for phonetic analysis.

3.1. Berkeley Common Environment

The BPM derives from another UC Berkeley project, BCE [4, 5], which has similar design goals for scientific research of all types on the UC Berkeley campus. As such, much of the core data analysis stack is provided by the BCE image and is found in various campus contexts—in workshops at the social sciences Data Lab, in the instructional lab at the Statistical Computing Facility, and at consultations with Berkeley Research Computing. The sharing of core functions with BCE means that students can pick up general purpose computing and data analysis skills from a variety of campus sources and be well-equipped to use them immediately for phonetic research. The core environment is quite literally the same in BCE and BPM, which means that users can focus on research questions rather than the computing environment. Little time is wasted dealing with software installation and configuration issues or with adjusting to user interface differences among computing environments.

3.1.1. BCE desktop

BCE provides a *standard reference* user environment. Its goal is to make it easy to provide step-by-step instructions to another user that are almost guaranteed to work since each user's operating environment is the same. Simple design choices are designed to reduce complexity: a solid color backdrop for the desktop; an icon for the Terminal next to the Applications menu; preconfigured space for tab-stops in the `nano` editor. Other choices are designed to affect performance, e.g. numeric libraries are carefully selected and preconfigured.

3.1.2. Python and R

Python and R packages are also selected, compiled, and installed by the BCE maintainers. In some cases the native Ubuntu packages are installed. Some of the BCE user community requires access to cutting edge software, in which case the maintainers install more up-to-date Python and R packages than are found in the official repositories. The BPM does not usually require these advanced capabilities but inherits them anyway; fortunately, we have not found that they introduce any incompatibilities with the software that we provision.

3.2. BPM additions

In addition to the basic data analysis software provided in BCE, the BPM includes a variety of software tools for phonetic research.

3.2.1. OpenSesame

OpenSesame [6] is a Python-based tool for creating and running psychology experiments. In the Phonlab we use this tool as a platform for running speech perception experiments. The OpenSesame version and installed plugins in the BPM are exactly matched to the Phonlab Linux workstations where researchers run their studies.

3.2.2. Penn Phonetics Lab Forced Aligner

The Penn Phonetics Lab Forced Aligner (P2FA) [7, 8] is a tool for automatic phonetic alignment of words and phones based on a transcript of a recording and using HTK (see section 4.0.1). P2FA contains acoustic models trained on American English and the `align.py` Python script, which prepares audio files and aligns them with a transcript by applying the acoustic models.

3.2.3. Praat and Wavesurfer

Praat [9] and Wavesurfer [10] are GUI tools for phonetic analysis. These are useful for waveform and spectrogram viewing, and for creating annotations for feature extraction and analysis via scripts. These scripts can be written in Praat's built-in scripting language, but in the Phonlab scripts are usually written in Python and employ BPM command-line tools.

3.2.4. ESPS

The Entropic Signal Processing System (ESPS) contains a variety of scriptable utilities for acoustic analysis, including `formant`, `melspec`, and `get_f0`. The ESPS utilities have been unmaintained since Microsoft bought Entropic in 1999, and compiling the code released by Microsoft in 64-bit environments is a problem. The Phonlab maintains a repository [11] with modifications for 64-bit compilation, and the compiled versions are distributed in the BPM.

3.2.5. ffmpeg

The `ffmpeg` utility [12] is a tool for working with audio and video materials. In the Phonlab it is used to automate the creation of video stimulus files—adding features such as lead-in still frames or replacing the audio track with processed audio.

3.2.6. ifcformant

The `ifcformant` command provides accurate formant measurements using the Inverse Filter Control algorithm [13], as well as F0 and RMS measurements.

3.2.7. klsyn

The klsyn package [14] includes Dennis Klatt's speech synthesizer [15], written in C and also provided with a Python API. The package also includes utilities for synthesizing speech from input parameter files, either as a single output (`klattsyn.py`) or as interpolated continua between two parameter sets (`klp_continuum.py`). The `klattsyn_interactive.py` script produces audio and parameter files in an interactive user session.

3.2.8. sox

The `sox` utility [16] is a tool for manipulating audio files that can easily be incorporated into scripts. It can be used to change file formats and sample rate, or it can filter and scale signals. Extracting portions of an audio file and concatenating multiple files are other common operations for this tool in the Phonlab.

3.2.9. audiolabel

The audiolabel package is a Python library for reading and processing phonetic label files, including Praat textgrids, and ESPS and Wavesurfer label files [17]. This package is frequently used by Phonlab researchers, who create textgrids by hand labeling or forced alignment with P2FA.

3.2.10. ultratils

The ultratils package [18] is a Python library under development in the Phonlab for ultrasound image extraction and analysis suitable for use with the Phonlab's Ultrasonix ultrasound system. One of its major utilities is the `psync` script for synchronizing audio files with ultrasound image data.

3.2.11. EdgeTrak

EdgeTrak is a Windows application for tracing and analyzing tongue contours in ultrasound images. We provide an install target for this software, but since it is quite specialized and requires Wine it is not installed by default.

3.2.12. creak_detect

The `creak_detect` tool is a precompiled Matlab function for detecting creaky voice regions from speech signals. It is part of the Voice Analysis Toolkit [19], a set of functions for glottal source and voice quality analysis.

4. Proprietary software

Some of the software used in the Phonlab requires the user to accept a license agreement and can therefore not be distributed in the BPM public image. These must be installed by the user, and the `bpm-update` command (see section 5.2) provides installation targets to assist users with these tasks.

4.0.1. Hidden Markov Model Toolkit

The Hidden Markov Model Toolkit (HTK) [20] provides utilities for speech recognition and is required by P2FA. This package is free to use but requires users to agree to a license before downloading. The `htk` target to `bpm-update` prompts for the user's registration details and then downloads, patches, and installs HTK.

4.0.2. Matlab runtime

The Phonlab does not actively develop Matlab programs, but there are useful programs and toolboxes that are freely available and might be distributed in the BPM in the future. One such example is the Voice Analysis Toolkit [19], and we are experimenting with providing a compiled version of one of its functions, which will require user installation of the Matlab runtime environment. The `creak_detect` target to `bpm-update` installs the compiled function.

4.0.3. Wine

The Wine package provides the capability of running 32-bit Windows applications in a Linux environment and is required for EdgeTrak (see section 3.2.11). Installation of Wine requires a user license agreement (indirectly, through a dependency), and a `wine` target to `bpm-update` is provided in the BPM.

5. Provisioning and maintenance

To produce the BPM image we first import a BCE virtual machine and install the BPM additions described in section 3.2. The resulting machine image is distributed publicly on the Phonlab web site [21].

5.1. BCE

The BCE maintainers produce a ready-made image for users and also share the Packer [22] recipe for creating the image as well as the Bash script that does system setup [23]. To create the BPM we start with the machine image.

5.2. `bpm-update`

The BPM additions to BCE are installed with the `bpm-update` Bash script [24]. This script contains installation targets that install one or more phonetic software packages. The `bpm-public` target is a master target and installs many of the other targets; the resulting machine image is exported and publicly distributed. `bpm-update` is available for users to install additional software as needed, and it is the primary way to distribute bug fixes to users. The `bpm-update` target updates `bpm-update` itself, which allows us to add new targets for software fixes and distribute them to users rapidly.

6. Discussion

The most common difficulty we have encountered is the initial installation of the virtual machine. The VirtualBox software itself installs easily, but importing and running the virtual machine can be a challenge. Users of Mac OS X systems have the least difficulty because virtualization hardware support is present and available on these systems. Windows systems are more diverse and virtualization support varies by manufacturer. Most Windows machines have hardware support for virtualization, but support for virtualization is not enabled by default on many of them. Users of these systems must enter the computer's BIOS setup to enable virtualization, and this is an unfamiliar task for most users. We usually dedicate an hour of class time at the beginning of the semester to assist users in installing and testing for virtual machine support.

Our experience with the BPM is that it has been of most use for Phonlab faculty and graduate students. Most of our users do not have a programming background and are not already familiar with Linux when they arrive in the Phonlab. As a result, there is a lot for a new user of the BPM to learn—how to navigate the Bash command shell, how to write Python scripts, and how to use the phonetic tools themselves. Phonlab graduate students are long-term users of the BPM and therefore have time to learn these skills. In addition to the experimental phonetics course, graduate students use the machine for their basic research, and the availability of software workshops that use BCE also provides opportunities for mastery of the BPM environment.

Undergraduates are exposed to the BPM primarily in one course, and they do not master all of the elements of the BPM. The long-term benefit to these users is therefore smaller, but they benefit nevertheless. The challenge of gaining these new skills is large regardless of the platform on which they learn, and the fact that all of the students use the same platform simplifies the instructional process. Students can be provided scripts for performing a phonetic analysis and an explicit set of steps for executing them. Before the use of the BPM these steps had to be given separately for the different operating environments found on the students' various computers, and much class time was wasted explaining the quirks of Windows vs. OS X vs. Linux environments. Additional time was wasted handling installation and execution issues on multiple versions of each of these environments.

One benefit of the BPM has been the ability to share fixes with users quickly and easily. For example, we have encountered multiple bugs in the P2FA `align.py` script, which we have fixed and made available to BPM users with a `bpm-update` target. The commands for applying the updates are easily shared with users since the machine environment is consistent.

Currently the BPM runs on ordinary host computers, but in the future we expect it could be available via cloud services and in high performance cluster environments. The BCE maintainers already produce images for cloud services like Amazon's EC2 and the UC Berkeley campus cluster, and these images are as consistent as possible with the images produced for ordinary hosts. We believe that the BPM additions will install easily on these images, which opens up the possibility of performing phonetic analysis on very large speech corpora. The ability of the BPM environment to scale in this way could be a benefit to users in the future. Development could proceed on a small sys-

tem with a subset of data and then run on larger systems only when needed. This strategy helps keep costs down since there is usually an expense related to large systems, and the consistency of the environment means that retooling existing workflows is not necessary when a project scales up in size.

6.1. Issues with the BPM

While our experience with the BPM has been largely positive, the use of virtual machines has challenges.

The first difficulty has been that some user machines are incapable of running virtual machines. Out of approximately 30 students who have enrolled in experimental phonetics in 2015 and 2016, one student's CPU lacked the hardware virtualization feature required to run the BPM. Another student's computer provided virtualization support but did not have sufficient RAM to run the guest operating system. Several other students were able to run the BPM but had underpowered hardware, and performance suffered; some users can boot the BPM in less than 15 seconds while others require two minutes or more.

A second but related problem has been with student computers that lack sufficient disk space to hold one or more machine instances. We recommend 14GB minimum for a single machine. This problem has caused a delay for some users, but in most cases the students in this situation have been able to clear sufficient space to complete the install.

Computer performance continues to increase, and it is possible that issues with slow-running and non-functional virtual machines will decrease over time as students come equipped with more powerful computers. Other computing innovations could disrupt this trend, however. As consumers move toward cloud- and app-based services they may migrate toward lower-powered systems (e.g. Chromebooks) or systems for which the keyboard-and-mouse desktop interface employed in the BPM is an uneasy match (e.g. tablets and mobile devices with touch-based interfaces). We will closely monitor the evolution of these trends in future years.

6.2. External use

Some of the choices made in assembling the BPM have been driven by concerns for training and support in the local computing context at UC Berkeley, but these choices do not prevent the BPM from being useful to phoneticians and other speech researchers outside of that context. External users of the BPM can benefit from a variety of ready-to-use analysis programs that are otherwise unavailable or difficult to install, particularly on host platforms where development environments are not normally available by default (Windows). The `ifcformant` and ESPS utilities fall into this category, as does HTK.

The general idea that a standardized computing platform with a specialized data analysis stack can benefit a group of users, either in the classroom or the research lab, is also not particular to our local context. External users can create their own local context for training and support that makes use of the BPM (or a locally-produced alternative).

While the BPM has been available for download on the Phonlab web site [21] since spring 2015, we have not tracked its usage and have not received any feedback from external users to date.

7. Conclusion

We have described the BPM, a virtual machine image with tools for phonetic research, and have shared our experience with using the machine in the Phonlab and related phonetics courses.

8. References

[1] *Ubuntu Server Guide*, 2014, https://help.ubuntu.com/14.04/serverguide/serverguide.pdf.

[2] [Online]. Available: https://www.virtualbox.org/

[3] Y. Yao, S. Tilsen, R. L. Sprouse, and K. Johnson, "Automated measurement of vowel formants in the Buckeye Corpus," *Journal of the Linguistic Society of Japan*, vol. 138, pp. 99–113, 2010.

[4] D. Clark, A. Culich, B. Hamlin, and R. Lovett, "BCE: Berkeley's common scientific compute environment for research and education," in *SCIPY 2014 – 13th Python in Science Conference, July 612, Austin, Texas, USA, Proceedings*, 2014, pp. 5–13.

[5] [Online]. Available: http://bce.berkeley.edu

[6] S. Mathôt, D. Schreij, and J. Theeuwes, "OpenSesame: An open-source, graphical experiment builder for the social sciences," *Behav Res*, vol. 44, no. 2, pp. 314–324, Nov 2011. [Online]. Available: http://dx.doi.org/10.3758/s13428-011-0168-7

[7] M. L. Jiahong Yuan, "Speaker identification on the SCOTUS corpus," in *Proceedings of Acoustics*, 2008, pp. 5687–5690.

[8] [Online]. Available: https://www.ling.upenn.edu/phonetics/old_website_2015/p2fa/

[9] [Online]. Available: http://www.fon.hum.uva.nl/praat/

[10] [Online]. Available: https://sourceforge.net/projects/wavesurfer/

[11] [Online]. Available: https://github.com/rsprouse/espsfree

[12] [Online]. Available: http://ffmpeg.org

[13] Y. Ueda, T. Hamakawa, T. Sakata, S. H. Hario, and A. Watanabe, "A real-time formant tracker based on the inverse filter control method," *Acoustical Science and Technology of the Acoustical Society of Japan*, vol. 28, no. 4, pp. 317–326, 2007.

[14] [Online]. Available: https://github.com/rsprouse/klsyn

[15] D. Klatt, "Software for a cascade/parallel formant synthesizer," *Journal of the Acoustic Society of America*, vol. 67, pp. 971–995, 1980.

[16] [Online]. Available: http://sox.sourceforge.net

[17] [Online]. Available: https://github.com/rsprouse/audiolabel

[18] [Online]. Available: https://github.com/rsprouse/ultratils

[19] [Online]. Available: https://github.com/jckane/Voice_Analysis_Toolkit

[20] [Online]. Available: http://htk.eng.cam.ac.uk/

[21] [Online]. Available: http://linguistics.berkeley.edu/plab/guestwiki/index.php?title=Berkeley_Phonetics_Machine

[22] [Online]. Available: https://www.packer.io/intro/

[23] [Online]. Available: https://github.com/ucberkeley/bce

[24] [Online]. Available: https://github.com/rsprouse/ucblingmisc/blob/master/bpm-bce/bpm-update-2015-spring