Nkululeko 1.0: A Python package to predict speaker characteristics with a high-level interface

Felix Burkhardt [1,2*] **and Bagus Tris Atmaja** [3*]

1 audEERING GmbH, Germany 2 TU Berlin, Germany 3 National Institute of Advanced Industrial Science and Technology (AIST), Japan * These authors contributed equally.

Summary

Nkululeko (Burkhardt, Wagner, et al., 2022) is open-source software written in Python and hosted on GitHub. It is predominantly a framework for audio-based machine learning explorations without the need to write Python code, and is strongly based on machine learning packages like sklearn (Pedregosa et al., 2011) and pytorch (Chaudhary et al., 2020). The main features are: training and evaluation of labelled speech databases with state-of-the-art machine learning approach and acoustic feature extractors, a live demonstration interface, and the possibility to store databases with predicted labels. Based on this, the framework can also be used to check on bias in databases by exploring correlations of target labels, like, e.g. depression or diagnosis, with predicted, or additionally given, labels like age, gender, signal distortion ratio or mean opinion score.

Design choices

The program is intended for **novice** people interested in speaker characteristics detection (e.g., emotion, age, and gender) without proficient in (Python) programming language. Its main target is for **education** and **research** with the main features as follows:

- Finding good combinations of variables, e.g., acoustic features, models (classifier or regressor), feature standardization, augmentation, etc., for speaker characteristics detection (e.g., emotion);

- Characteristics of the database, such as distribution of gender, age, emotion, duration, data size, and so on with their visualization;

- Inference of speaker characteristics from a given audio file or streaming audio (can be said also as "weak" labeling for semi-supervised learning).

Hence, one should be able to use Nkululeko after installing and preparing/downloading their data in the correct format in a single line.

```
$ nkululeko.MODULE_NAME --config CONFIG_FILE.ini
```

How does it work?

nkululeko is a command line tool written in Python, best used in conjunction with the Visual Studio Code editor (but can be run stand-alone). To use it, a text editor is needed to edit the experiment configuration. You would then run nkululeko like this:

```
$ nkululeko.explore --config conf.ini
```

34 and inspect the results afterward; they are represented as images, texts, and even a fully
35 automatically compiled PDF report written in latex.

36 nkululeko's data import format is based on a simple CSV formalism, or alternatively, for
37 a more detailed representation including data schemata, audformat.[1] Basically, to be used
38 by nkululeko, the data format should include the audio file path and a task-specific label.
39 Optionally, speaker ID and gender labels help with speech data. An example of a database
40 labelled with emotion is

```
file, speaker, gender, emotion
x/sample.wav, s1, female, happy
...
```

41 As the main goal of nkululeko is to avoid the need to learn programming, experiments are
42 specified by means of a configuration file. The functionality is encapsulated by software
43 *modules* (interfaces) that are to be called on the command line. We list the most important
44 ones here:

- 45 **nkululeko**: do machine learning experiments combining features and learners
- 46 **demo**: demo the current best model on the command line or some files
- 47 **test**: run the current best model on a specified test set
- 48 **explore**: perform data exploration (used mainly in this paper)
- 49 **augment**: augment the current training data. This could also be used to reduce bias
- 50 in the data, for example, by adding noise to audio samples that belong to a specific
- 51 category.
- 52 **aug_train**: augment the training data and train the model with the augmented data.
- 53 **predict**: predict features like speaker diarization, signal distortion ratio, mean opinion
- 54 score, arousal/valence, age/gender (for databases that miss this information), with deep
- 55 neural nets models, e.g. as a basis for the *explore* module.
- 56 **segment**: segment a database based on VAD (voice activity detection)
- 57 **ensemble**: ensemble several models to improve performance

58 The configuration (INI) file consists of a set of key-value pairs that are organised into several
59 sections. Almost all keys have default values, so they do not have to be specified.

60 Here is a sample listing of an INI file (conf.ini) with a database section:

```ini
[EXP]
name = explore-androids
[DATA]
databases = ['androids']
androids = /data/androids/androids.csv
target = depression
labels = ['depressed', 'control']
samples_per_speaker = 20
min_length = 2
[PREDICT]
sample_selection = all
targets = ['pesq', 'sdr', 'stoi', 'mos']
[EXPL]
value_counts = [['gender'], ['age'], ['est_sdr'], ['est_pesq'], ['est_mos']]
[REPORT]
latex = androids-report
```

61 As can be seen, some of the values simply contain Python data structures like arrays or
62 dictionaries. Within this example, an experiment is specified with the name *explore-androids*,

---

[1] https://audeering.github.io/audformat/

and a result folder with this name will be created, containing all figures and textual results, including an automatically generated Latex and PDF report on the findings.

The *DATA* section sets the location of the database and specifies filters on the sample, in this case limiting the data to 20 samples per speaker at most and at least 2 seconds long. In this section, the split sets (training, development, and test) are also specified. There is a special feature named *balance splits* that lets the user specify criteria that should be used to stratify the splits, for example, based on signal distortion ratio.

With the *predict* module, specific features like, for example, signal distortion ratio or mean opinion score are to be predicted by deep learning models. The results are then used by a following call to the *explore* module to check whether these features, as well as some ground truth features (*age* and *gender*), correlate with the target variable (*depressed* in the given example) in any way.

The `nkululeko` configuration can specify further sections:

- **FEATS** to specify acoustic features (e.g. opensmile (Eyben et al., 2010) or deep learning embeddings; e.g. wav2vec 2.0 (Baevski et al., 2020)) that should be used to represent the audio files.
- **MODEL** to specify statistical models for regression or classification of audio data.

Example of usage

In the previous section, we have seen how to specify an experiment in an INI file which can be run with, for instance, explore and segment modules. Here, we show how to run the experiment (`nkululeko.nkululeko`) with built-in dataset (Polish Speech Emotions dataset) from the installation until getting the results.

First, novices could clone the GitHub repository of nkululeko.

```
$ git clone https://github.com/felixbur/nkululeko.git
$ cd nkululeko
```

Then, install nkululeko with `pip`. It is recommended that a virtual environment be used to avoid conflicts with other Python packages.

```
$ python -m venv .env
$ source .env/bin/activate
$ pip install nkululeko
```

Next, extract `polish_speech_emotions.zip` inside the nkululeko data folder (`nkululeko/data/polish`) with right click regardless of the operating system (or using `unzip` command in the terminal like below). Then, run the following command in the terminal:

```
$ cd data/polish
$ unzip polish_speech_emotions.zip
$ python3 process_database.py
$ cd ../..
$ nkululeko.nkululeko --config data/polish/exp.ini
```

That's it! The results will be stored in the `results/exp_polish_os` folder as stated in `exp.ini`. Below is an example of the debug output of the command:

```
DEBUG: nkululeko: running exp_polish_os from config data/polish/exp.ini,
nkululeko version 0.91.0
...
DEBUG: reporter:
                precision    recall  f1-score   support
```

```
           anger    0.6944   0.8333   0.7576      30
         neutral    0.5000   0.4333   0.4643      30
            fear    0.6429   0.6000   0.6207      30

        accuracy                      0.6222      90
       macro avg    0.6124   0.6222   0.6142      90
    weighted avg    0.6124   0.6222   0.6142      90

DEBUG: reporter: labels: ['anger', 'neutral', 'fear']
DEBUG: reporter: result per class (F1 score): [0.758, 0.464, 0.621]
from epoch: 0
DEBUG: experiment: Done, used 7.439 seconds
DONE
```

Statement of need

Open-source tools are believed to be one of the reasons for accelerated science and technology. They are more secure, easy to customise, and transparent. There are several open-source tools that exist for acoustic, sound, and audio analysis, such as librosa (McFee et al., 2015), TorchAudio (Yang et al., 2021), pyAudioAnalysis (Giannakopoulos, 2015), ESPNET (Watanabe et al., 2018), and SpeechBrain (Ravanelli et al., 2021). However, none of them are specialised in speech analysis with high-level interfaces for novices in the speech processing area.

One exception is Spotlight (Suwelack, 2023), an open-source tool that visualises metadata distributions in audio data. An existing interface between nkululeko and Spotlight can be used to combine the visualisations of Spotlight with the functionalities of Nkululeko.

Nkululeko follows these principles:

- *Minimum programming skills*: The only programming skills required are preparing the data in the correct (CSV) format and running the command line tool. For AUDFORMAT, no preparation is needed.

- *Standardised data format and label*: The data format is based on CSV and AUDFORMAT, which are widely used formats for data exchange. The standard headers are like 'file', 'speaker', 'emotion', 'age', and 'language' but also can be customised. Data could be saved anywhere on the computer, but the recipe for the data preparation is advised to be saved in nkululeko/data folder (and/or make a soft link to the original data location).

- *Replicability*: the experiments are specified in a configuration file, which can be shared with others including the splitting of training, development, and test partition. All results are stored in a folder with the same name as the experiment.

- *High-level interface*: the user specifies the experiment in an INI file, which is a simple text file that can be edited with any text editor. The user does not need to write Python code for experiments.

- *Transparency*: as CLI, nkululeko *always output debug*, in which info, warning, and error will be obviously displayed in the terminal (and should be easily understood). The results are stored in the experiment folder for further investigations and are represented as images, texts, and even a fully automatically compiled PDF report written in latex.

Usage in existing research

Nkululeko has been used in several research projects since its first appearance in 2022 (Burkhardt, Wagner, et al., 2022). The following list gives an overview of the research papers that have used Nkululeko:

- (Burkhardt, Eyben, et al., 2022): this paper reported a database development of synthesized speech for basic emotions and its evaluation using the Nkululeko toolkit.
- (Burkhardt et al., 2024): this paper shows how to use Nkululeko for bias detection. The findings on two datasets, UACorpus and Androids, show that some features are correlated with the target label, e.g., depression, and can be used to detect bias in the database.
- (Atmaja et al., 2024): this paper shows Nkululeko's capability for ensemble learning with a focus on uncertainty estimation.
- (Atmaja & Sasou, 2025): in this paper, evaluations of different handcrafted acoustic features and SSL approaches for pathological voice detection tasks were reported, highlighting the ease of using Nkululeko to perform extensive experiments including combinations of different features at different levels (early and late fusions)

Acknowledgements

References

Atmaja, B. T., & Sasou, A. (2025). Pathological voice detection from sustained vowels: Handcrafted vs. Self-supervised learning. *2025 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)*.

Atmaja, B. T., Sasou, A., & Burkhardt, F. (2024). UNCERTAINTY-BASED ENSEMBLE LEARNING FOR SPEECH CLASSIFICATION. *Orient. COCOSDA*, 1–6.

Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, & H. Lin (Eds.), *Advances in neural information processing systems* (Vol. 33, pp. 12449–12460). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/file/92d1e1eb1cd6f9fba3227870bb6d7f07-Paper.pdf

Burkhardt, F., Atmaja, B. T., Derington, A., & Eyben, F. (2024). CHECK YOUR AUDIO DATA : NKULULEKO FOR BIAS DETECTION. *Orient. COCOSDA*, 1–6.

Burkhardt, F., Eyben, F., & Schuller, W. (2022). SyntAct : A Synthesized Database of Basic Emotions. In Jonne Sälevä & C. Lignos (Eds.), *Proc. Work. Dataset creat. Low. Lang. Within 13th lang. Resour. Eval. conf.* European Language Resources Association.

Burkhardt, F., Wagner, J., Wierstorf, H., Eyben, F., & Schuller, B. (2022). Nkululeko: A tool for rapid speaker characteristics detection. *2022 Language Resources and Evaluation Conference, LREC 2022*, 1925–1932. ISBN: 9791095546726

168 Chaudhary, A., Chouhan, K. S., Gajrani, J., & Sharma, B. (2020). *Deep learning with PyTorch*.
169 https://doi.org/10.4018/978-1-7998-3095-5.ch003

170 Eyben, F., Wöllmer, M., & Schuller, B. (2010). openSMILE – the munich versatile and fast
171 open-source audio feature extractor. *MM'10 - Proceedings of the ACM Multimedia 2010*
172 *International Conference*, 1459–1462. https://doi.org/10.1145/1873951.1874246

173 Giannakopoulos, T. (2015). pyAudioAnalysis: An open-source python library for audio signal
174 analysis. *PLoS One*, *10*(12), 1–17. https://doi.org/10.1371/journal.pone.0144610

175 McFee, B., Raffel, C., Liang, D., Ellis, D., McVicar, M., Battenberg, E., & Nieto, O. (2015).
176 librosa: Audio and Music Signal Analysis in Python. *Proc. 14th Python Sci. Conf.*, *Scipy*,
177 18–24. https://doi.org/10.25080/majora-7b98e3ed-003

178 Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M.,
179 Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D.,
180 Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python.
181 *Journal of Machine Learning Research*, *12*, 2825–2830.

182 Ravanelli, M., Parcollet, T., Plantinga, P., Rouhe, A., Cornell, S., Lugosch, L., Subakan, C.,
183 Dawalatabad, N., Heba, A., Zhong, J., Chou, J.-C., Yeh, S.-L., Fu, S.-W., Liao, C.-F.,
184 Rastorgueva, E., Grondin, F., Aris, W., Na, H., Gao, Y., … Bengio, Y. (2021). *SpeechBrain:*
185 *A general-purpose speech toolkit*. https://arxiv.org/abs/2106.04624

186 Suwelack, S. (2023). Spotlight. In *GitHub repository*. https://github.com/Renumics/
187 spotlight/; GitHub.

188 Watanabe, S., Hori, T., Karita, S., Hayashi, T., Nishitoba, J., Unno, Y., Soplin, N. E. Y.,
189 Heymann, J., Wiesner, M., Chen, N., Renduchintala, A., & Ochiai, T. (2018). ESPNet:
190 End-to-end speech processing toolkit. *Proc. Annu. Conf. Int. Speech Commun. As-*
191 *soc. INTERSPEECH*, *2018-Septe*(September), 2207–2211. https://doi.org/10.21437/
192 Interspeech.2018-1456

193 Yang, S., Chi, P.-H., Chuang, Y.-S., Lai, C.-I. J., Lakhotia, K., Lin, Y. Y., Liu, A. T., Shi,
194 J., Chang, X., Lin, G.-T., Huang, T.-H., Tseng, W.-C., Lee, K., Liu, D.-R., Huang,
195 Z., Dong, S., Li, S.-W., Watanabe, S., Mohamed, A., & Lee, H. (2021). SUPERB:
196 Speech Processing Universal PERformance Benchmark. *Interspeech 2021*, 1194–1198.
197 https://doi.org/10.21437/Interspeech.2021-1775