Patrick Bos

18/11/2022

General impression

Overall, this is a really great piece of software, so my compliments!

Overall impression: apart from a few bugs, the demos and general usage seems very straightforward. Just get a video file and apply a pipeline. Well documented from a user perspective. Great experience!

As you yourself also suspected, downloading all the assets is a bit of a pain. I don't see an obvious way around it though, so as long as you have good expectation management (i.e. good progress bars) it shouldn't hurt the user experience too much.

Package generally looks very good. It was relatively easy to find things. Very extensive testing, great!

----------------------------------------------------------------

1. Install instructions are very verbose, which is probably great for beginners, but maybe better suited in a separate beginner's guide. For instance: creating a virtual environment is basic Python nowadays. I do think it's very valuable that you reiterate it, but I just wouldn't do it in the README.md on GitHub.

*Easy fix. We create two subsections in the installation part: (Quick installation) vs (detailed installation instructions). I wouldn’t use the term ‘beginner’s guide’ as this conveys a judgment, even if it’s involuntary.*

1. The Conda instructions don't work like this. When you create the environment, you also have to install Python in it, otherwise it will still just use your system Python. So change to: `conda create -n mexca-venv "python<3.10"`.

*Easy fix. I would reiterate the same concept with virtualenv to make it consistent: i.e., python3.10 -m venv*

1. I love the logo on the front page (actually the whole README.md looks really good), but it would be even more helpful if you added a screenshot in the How to Use Mexca section.

*Won’t fix. Not sure I understand here which screenshot would like us to have? If it’s about the mexca’s output, we have already tried, and it did not look well.*

1. The Getting Started section suggests to try out the demo notebook. It would be really cool if people can immediately try that out themselves without having to install stuff. You could use <https://mybinder.org/> for that, or, if you’re feeling adventurous and cutting-edge, try to set up jupyterlite (a webassembly of version of jupyter, i.e. it runs fully in the browser, without needing a regular Python process in the background!) to run from Github pages.

*Major fix. We thought about having google colab for this purpose. Unfortunately, it doesn’t work with mexca’s python version, but these two solutions sounds good. This point might require some degree of exploration (as per using jupyterlite or binder) but I think it’s important to solve it as per his suggestion.*

1. I love that you also mention in Intended use the privacy aspects. Very thorough! The whole readthedocs docs are thorough and clearly written, actually.
2. Getting started with mexca: Do I need a read or a write authentication token? (I picked read)

*Easy fix. Just leave instructions on which one to use (i.e., read, as per huggingface* [*instructions*](https://huggingface.co/docs/hub/security-tokens)*)*

1. Do I “Add token as git credential?” I chose yes, but then got lots of red scary text. I guess it still worked, but I should have picked no?

*No Idea.*

1. Oops! `from mexca.core.pipeline import Pipeline` took a few seconds and then produced a long error ending in: `DistributionNotFound: The 'wandb' distribution was not found and is required by the application`. Weirdly, wandb *\*was\** importable, even though it was not installed in the environment at that point. Strange! The fix for me was to install wandb, so probably you should add that to your dependencies. It's weird though that this doesn't fail in your testcases where the same `Pipeline` import is done, e.g. in tests/core/test\_pipeline.py...

*Won’t fix. I am not sure why this happens to him – I would not change anything because we can’t replicate this behaviour, but I would still keep it in mind.*

1. To do the demo I cloned the repo and opened the notebook there. The first cell complains about ipywidgets missing for tqdm's IProgress thingy:[ /Users/pbos/sw/miniconda3/envs/mexca/lib/python3.9/site-packages/tqdm/auto.py:22: TqdmWarning: IProgress not found. Please update jupyter and ipywidgets. See https://ipywidgets.readthedocs.io/en/stable/user\_install.html

from .autonotebook import tqdm as notebook\_tqdm] Not a breaking thing, but would be nice to fix this. You could add a "demo" dependencies section in setup.cfg and add `python3 -m pip install .[demo]` as install instructions. Actually, in cell 4 the missing ipywidgets do break the demo, so indeed it's a necessary addition. I did `pip install ipywidgets` to quickly fix it.

Easy fix.

1. Btw, it's possible to run a test suite for your demo notebooks in CI, so that you can be sure they won't break. Check out the DIANNA repo for an example workflow: <https://github.com/dianna-ai/dianna/blob/main/.github/workflows/notebooks.yml>

*Easy fix, but see below.*

1. Cell 5, after 13 progress bars, failed with `ValueError: Argument "num\_speakers" must be >= 2 for speaker identification`. Fixed by prepending in core/pipeline.py on line 142 an argument `None` before `use\_auth\_token`.

*Major fix. We need to add automated testing, if not of the whole pipeline in the demo, at least of the imports of the libraries and other small things*

1. After I did this, Cell 5 continued with another bunch of progress bars :) Most of these do have M/k/G units, so that's great. Maybe also put a big fat warning in the Markdown cell about cell 5 saying that this step will take long the first time (at least 15 minutes for me, depends on connection speed of course).

*Easy fix.*

1. This probably also makes the demo unsuitable for automated testing btw (as suggested above). If simpler demos can be thought of (well, not simpler, just with less GBs to download), that would probably be better overall (as you indeed already indicated yourselves).
2. In parallel, while waiting for the demo notebook to run, I also started the example\_emotion\_feature\_extraction notebook. It failed after the second progress bar with `HTTPError: HTTP Error 404: Not Found` originating from:

File ~/sw/miniconda3/envs/mexca/lib/python3.9/site-packages/mexca/video/extraction.py:52, in FaceExtractor.\_\_init\_\_(self, au\_model, landmark\_model, \*\*clargs)

48 self.resnet = InceptionResnetV1(

49 pretrained='vggface2'

50 ).eval()

51 self.cluster = SpectralClusterer(\*\*clargs)

---> 52 self.pyfeat = feat.detector.Detector(

53 au\_model=au\_model,

54 landmark\_model=landmark\_model

55 )

```

Major fix. This is connected to point 11.

1. I would advise to integrate the several downloading bars into just one overall bar if that is possible.

*Easy fix.*

1. To make things even easier to find (for me at least), you could consider simplifying the file structure a little. For instance, imho, it doesn't really add much to have a separate file (== submodule) for each class. `AudioIntegrator` is the only thing in audio/integration.py, `SpeakerIdentifier` is the only thing in audio/identification.py, etcetera. You could also just make one audio.py file containing all 4 audio classes. This would save some importing and make things a bit easier to find. Same goes for core, video and text. Again, this is just my opinion, FYC, ymmv.

Easy fix.

1. Note that empty `\_\_init\_\_.py` files are no longer necessary (since Python 3.3). If you remove them you do have to change the setup.py/setup.cfg, see <https://packaging.python.org/en/latest/guides/packaging-namespace-packages/#native-namespace-packages>

*Easy fix*

1. Maybe you can run tests in parallel to speed up CI. You could just create separate workflow jobs for separate tests. I have to say that it doesn't look as bad as you said, I only see a 22 minute macOS run that I'd try to get down to a few minutes, but otherwise seems workable to me.

*Parallelise testing. Patrick didn’t notice the commented tests (weird) but his suggestions may well fix the timing issue.*

1. Another option could be optimizing pip install, which also takes 4-8 minutes I see. You could try a few things, like switching to mamba (a faster conda alternative), to see if that speeds up installation. You also just have a lot of dependencies. Maybe you can filter out the ones you don't *\*absolutely\** need for running some portion of the tests.

*Patrick here slightly touches on the problem of the numerous dependencies. I agree to double check* *that they are essential. Maybe there is an automated way to do this?*

1. I think you could also see a speedup if you `pip install wheel` before installing any dependencies. Just add it to the first line, `python3 -m pip install --upgrade pip setuptools wheel`.

*Easy fix.*

1. I commented on this here and there while trying it out. Indeed, as you mentioned, you'll need to profile your code if you want to improve it. Having briefly looked at it, I don't expect major opportunities on your side; probably most of the code is simply busy executing Hugging Face model evaluations.

But you should still profile to make sure!

If/when you do, I'd be happy to provide some tips to get started if you need them.

*Major fix.*