Workshop_Linguistic Speech Tracking.ipynb - Colaboratory https://colab.research.google.com/drive/1TenZTwYLW...

Install necessary packages

Show time (installation can take around 10 minutes)

[ ] ↳ 1 cell hidden

Install miniconda and mamba

```
!pip install -q condacolab
import condacolab
condacolab.install()
```

```
⬇️ Downloading https://github.com/conda-forge/miniforge/releases/download/2
📦 Installing...
📌 Adjusting configuration...
🩹 Patching environment...
⏲ Done in 0:00:20
🔁 Restarting kernel...
```

Double-click (or enter) to edit

Install eelbrain

```
import condacolab
condacolab.check()
```

```
✨🍰✨ Everything looks OK!
```

```
!pip install praat-textgrids
```

```
Collecting praat-textgrids
  Downloading praat_textgrids-1.4.0-py3-none-any.whl (25 kB)
Installing collected packages: praat-textgrids
Successfully installed praat-textgrids-1.4.0
WARNING: Running pip as the 'root' user can result in broken permissions an
```

```
!mamba install -y eelbrain
```

Check if installation is successful

```
print(eelbrain.__version__)
```

```
0.39.7
```

Mount Google Drive

```
# mount google shared folder to your google drive:
# (1) open the link (https://drive.google.com/drive/folders/1gkbvzQzu-an_fwcj2fY
# (2) click on the small arrow next to 'Salzburg'
# (3) make shortcut to your own Drive
# aka no need to mount the google drive locally

from google.colab import drive
drive.mount('/mnt/drive')
```

```
Mounted at /mnt/drive
```

Setup

Initialize and add parameters

Import packages

```
import re

import eelbrain
import mne
import numpy as np

# eelbrain
import eelbrain
# see documentation of eelbrain toolbox: https://eelbrain.readthedocs.io

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os
import textgrids

from pathlib import Path
```

Data locations

Workshop_Linguistic Speech Tracking.ipynb - Colaboratory https://colab.research.google.com/drive/1TenZTwYLW...

```
# Location of data
DATA_ROOT = '/mnt/drive/MyDrive/CNSP_workshop'
STIMULI_DIR = DATA_ROOT + '/data/stimuli'
OTHER_DIR = DATA_ROOT + '/data/other'
TRF_DIR = DATA_ROOT + '/data/TRFs'
EEG_DIR = DATA_ROOT + '/data/eeg'
```

1_Making speech features

Crafting the speech features is the key of a good model. Depending on the research questions, different speech features should be used. Here, we are interested in linguistic speech tracking. However, since linguistic features are highly correlated with acoustic speech features, we want to control for acoustic speech processing. Hence, acoustic, (sub-)lexical as well as linguistic speech features should be extracted.

Acoustic speech features

```
# visualize the speech spectrogram
# the speech spectrogram was created using the gammatone filterbank, this quite
spectrogram = eelbrain.load.unpickle(os.path.join(STIMULI_DIR, 'audiobook_1_64Hz
p = eelbrain.plot.Array(spectrogram.sub(time=(97, 100)))
```

```
# Exercise: Create and Visualize the feature acoustic onsets. Acoustic onsets is
# To calculate acoustic onsets, the spectrogram is derived and halve-wave rectif
# These manipulation can be easily done using eelbrain, check out the class nd v
acoustic_onsets = spectrogram.diff('time').clip(0, name='acoustic_onsets')
p = eelbrain.plot.Array(acoustic_onsets.sub(time=(97, 100)))
```

```
# Exercise: Check out the relation between spectrogram and acoustic onsets (to d
#p = eelbrain.plot.UTS([[spectrogram.mean('frequency', name='avg_spectrogram').s
p = eelbrain.plot.UTS([[spectrogram.sub(frequency=124, name='spectrogram_L_freq'
```

(Sub-)lexical features

```
# To create the phoneme and word onsets features, we determined the onset of a w
# By uploading the wav file and the transcript, a textgrid is obtained. From thi
textgrid_file = os.path.join(OTHER_DIR, 'audiobook_1.TextGrid')
fs = 1/spectrogram.time.tstep # sampling rate by preprocessed EEG of sparrkulee
```

```
grid = textgrids.TextGrid(textgrid_file)
grid.keys()
```

```
odict_keys(['ORT-MAU', 'KAN-MAU', 'MAU'])
```

```
grid['ORT-MAU'][15]
```

```
<Interval text="maar" xmin=15.06 xmax=15.21>
```

```
word_onset_times = [interval.xmin for interval in grid['ORT-MAU'] if interval.te
```

```
time = eelbrain.UTS.from_int(0,  np.ceil(grid.xmax*fs), fs)
stimulus_wordOnsets = eelbrain.NDVar(np.zeros(len(time)), time, name='word onset
stimulus_wordOnsets[word_onset_times] = 1
```

```
# visualize the word onsets on the spectrogram (averaged across frequency bands)
p = eelbrain.plot.UTS([[spectrogram.mean('frequency', name='avg_spectrogram').su
```

```
# Exercise: make the stimulus for phoneme onsets and plot it together with the s
# A tip: unknown utterances are indicated by following phonemes: '<p:>', '<p>',
phonemes_onset_times = [interval.xmin for interval in grid['MAU'] if
                        interval.text not in ['<p:>', '<p>', '<nib>', '<usb>']]

stimulus_phonemeOnsets = eelbrain.NDVar(np.zeros(len(time)), time, name='phoneme
stimulus_phonemeOnsets[phonemes_onset_times] = 1
```

```
p = eelbrain.plot.UTS([[spectrogram.mean('frequency', name='avg_spectrogram').su
                        stimulus_wordOnsets.sub(time=(97, 100))]])
```

Linguistic features

```
# Linguistic features require language models (or at least word probabilities) a
phoneme_surprisal = eelbrain.load.unpickle(os.path.join(STIMULI_DIR, 'audiobook_
p = eelbrain.plot.UTS(phoneme_surprisal.sub(time=(97, 100)))
```

```
# Exercise: plot all linguistic speech features on top of each other
cohort_entropy = eelbrain.load.unpickle(os.path.join(STIMULI_DIR, 'audiobook_1_6
word_surprisal = eelbrain.load.unpickle(os.path.join(STIMULI_DIR, 'audiobook_1_6
word_frequency = eelbrain.load.unpickle(os.path.join(STIMULI_DIR, 'audiobook_1_6
```

```
p = eelbrain.plot.UTS([[phoneme_surprisal.sub(time=(97, 100)), cohort_entropy.su
                        word_surprisal.sub(time=(97, 100))+6, word_frequency.sub
```

2_Estimating forward models

```
# estimate the forward model using the spectrogram and inspect the prediction ac
# get the EEG → make it an eelbrain ND var
eeg_data = np.load(os.path.join(EEG_DIR, 'sub-001_ses-shortstories01_task-listen
sensor = eelbrain.Sensor.from_montage('biosemi64')[:64]
time = eelbrain.UTS(0, 1 / fs, eeg_data.shape[1])
eeg = eelbrain.NDVar(eeg_data.astype('float64'), (sensor, time), name='eeg_data'

eeg = eeg.sub(time=(None, spectrogram.time.tstop))


# estimate the forward model
# → time is reduced to 3min (180s) to speed up the computation time
r = eelbrain.boosting(eeg.sub(time=(None, 180)), spectrogram.sub(time=(None, 180


# visualize the prediction accuracy
p = eelbrain.plot.Topomap(r.r, head_radius=0.45, clip='circle')
c = p.plot_colorbar('Prediction accuracy [Pearson\'s r]')


# visualize the trf
plot = eelbrain.plot.TopoButterfly(r.h.mean('frequency'), t=0.11, ylabel='TRF we


# Exercise: check out the effect of setting a different basis kernel. This affec
# to reduce computation time: reduce to 1 minute and disable the test partition
# visualize the results for a channel selection (see below)
results = []
for basis in [0, 0.05, 0.1, 0.15]:
  r_b = eelbrain.boosting(eeg.sub(time=(None, 60)), spectrogram.sub(time=(None,
  results.append(r_b)


channel_selection = ['F1', 'Fz', 'F2', 'FC1', 'FCz', 'FC2']
sensors = eeg.get_dim('sensor')
zeros_nd = eelbrain.NDVar(np.zeros(64), (sensors,))

p = eelbrain.plot.Topomap(zeros_nd, head_radius=0.45, clip='circle')
p.mark_sensors(channel_selection, c='r')


p = eelbrain.plot.UTS([[r.h.mean('frequency').sub(sensor=channel_selection).mean


# Exercise: adding multiple features (e.g., spectrogram and phoneme onsets) and
# that we've created earlier)
r_2 = eelbrain.boosting(eeg.sub(time=(None, 180)), [spectrogram.sub(time=(None,


# visualize the prediction accuracy
p = eelbrain.plot.Topomap(r_2.r, head_radius=0.45, clip='circle')
c = p.plot_colorbar('Prediction accuracy [Pearson\'s r]')


# visualize the trf
```



```
plot = eelbrain.plot.TopoButterfly(r_2.h[0].mean('frequency'), t=0.11, ylabel='T
plot = eelbrain.plot.TopoButterfly(r_2.h[1], t=0.2, ylabel='TRF weights [a.u.]',


# Exercise: determine the added value of phoneme onsets on top of the spectrogra
r_added_value = r_2.r - r.r
p = eelbrain.plot.Topomap(r_added_value, head_radius=0.45, clip='circle', vmax=0
c = p.plot_colorbar('Prediction accuracy [Pearson\'s r]')


# You can get insights in how the data is split in order to estimate the forward
# specify the amount of partitions using partitions=5; testing partition test=Tr
p = eelbrain.plot.preview_partitions(partitions=5, test=True)
```

3_Added value of linguistic features

```
# read all the files, save into a eelbrain dataframe
files = os.listdir(TRF_DIR)

rows = []
for filename in files:
  d = eelbrain.load.unpickle(os.path.join(TRF_DIR, filename))

  subject = filename.split('_')[0]
  model = filename.split('_')[1].replace('.pickle', '')

  rows.append([subject, model, d.r])


# make dataframe
ds_prediction_accuracy = eelbrain.Dataset.from_caselist(['subject', 'model', 'pr


# visualize the prediction accuracy for each model
p = eelbrain.plot.Topomap('prediction_accuracy', 'model', ds=ds_prediction_accur
c = p.plot_colorbar('Prediction accuracy [Pearson\'s r]')


# check if there is a significant difference
res = eelbrain.testnd.TTestRelated('prediction_accuracy', 'model', match='subjec
p = eelbrain.plot.Topomap(res, clip='circle', head_radius=0.45, ncol=3)
c = p.plot_colorbar('Prediction accuracy [Pearson\'s r]')


p = eelbrain.plot.Topomap(res.masked_difference(), clip='circle', head_radius=0.
c = p.plot_colorbar('Prediction accuracy [Pearson\'s r]')


# Exercise: added value of multiple linguistic features, to do so, for each subj
rows_linguistic_tracking = []
for subject in np.unique(ds_prediction_accuracy['subject']):
  complete = ds_prediction_accuracy[np.logical_and(ds_prediction_accuracy['subje
  assert complete.shape[0]==1
```

```
  baseline = ds_prediction_accuracy[np.logical_and(ds_prediction_accuracy['subje
  assert baseline.shape[0]==1

  rows_linguistic_tracking.append([subject, complete - baseline])

# to dataframe
ds_linguistic_tracking = eelbrain.Dataset.from_caselist(['subject', 'linguistic_


# visualize linguistic tracking
p = eelbrain.plot.Topomap('linguistic_tracking', ds=ds_linguistic_tracking, ncol
c = p.plot_colorbar('Difference in Prediction accuracy [Pearson\'s r]')


# next up: look at the TRFs of the complete model.
# note: an eelbrain dataframe requires that the features are in the same dimensi
rows_trf = []
for filename in files:
  d = eelbrain.load.unpickle(os.path.join(TRF_DIR, filename))

  subject = filename.split('_')[0]
  model = filename.split('_')[1].replace('.pickle', '')

  if not model in ['complete']:
    continue

  trfs = d.h
  for trf in trfs:
    if trf.name in ['binned spectrogram', 'acoustic onsets']:
      trf = trf.mean('frequency')
    rows_trf.append([subject, trf, trf.name])


# to dataframe
ds_trf = eelbrain.Dataset.from_caselist(['subject', 'trf', 'feature'], rows_trf,


np.unique(ds_trf['feature'])

     array(['acoustic onsets', 'binned spectrogram', 'cohort_entropy_subtlex',
            'phoneme onsets', 'phoneme_surprisal_subtlex', 'word onsets',
            'word_frequency_ngram', 'word_surprisal_ngram'], dtype='<U25')


# visualize the TRF for spectrogram
p = eelbrain.plot.TopoButterfly('trf', ds=ds_trf[ds_trf['feature']=='binned spec


# take average across channel selection
trf_spectrogram_channel_selection = ds_trf[ds_trf['feature']=='binned spectrogra


p = eelbrain.plot.UTSStat(trf_spectrogram_channel_selection)


res = eelbrain.testnd.TTestOneSample(ds_trf[ds_trf['feature']=='binned spectrogr
```

Workshop_Linguistic Speech Tracking.ipynb - Colaboratory https://colab.research.google.com/drive/1TenZTwYLW...

```
p = eelbrain.plot.Butterfly(res)


# Exercise: visualize the TRFs of the linguistic features.
# Tip: before averaging across a channel selection, inspect the TRF across all c
# channel selection needs to be taken
for feature, t in zip(['cohort_entropy_subtlex', 'phoneme_surprisal_subtlex', 'w
                       [0.3, 0.25, 0.3, 0.35]):
  p = eelbrain.plot.TopoButterfly('trf', ds=ds_trf[ds_trf['feature']==feature],


for feature in ['cohort_entropy_subtlex', 'phoneme_surprisal_subtlex', 'word_fre
  res = eelbrain.testnd.TTestOneSample(ds_trf[ds_trf['feature']==feature, 'trf']
  p = eelbrain.plot.Butterfly(res, title=feature)


channel_selection = ['C1','Cz','C2', 'CP1', 'CPz', 'CP2']


for feature in ['cohort_entropy_subtlex', 'phoneme_surprisal_subtlex', 'word_fre
  trf = ds_trf[ds_trf['feature']==feature, 'trf'].sub(sensor=channel_selection).
  res = eelbrain.testnd.TTestOneSample(trf)
  plot = eelbrain.plot.UTSStat(trf, title=feature)

  # indicate the clusters
  for tstart, tstop, p in zip(res.clusters['tstart'], res.clusters['tstop'], res
    if p < 0.05:
      plot.add_vspan(tstart, tstop)



res.clusters
```

id	tstart	tstop	duration	p	sig
1	-0.015625	0.09375	0.10938	0	***
2	0.25	0.51562	0.26562	0	***

NDVars: cluster

Colab paid products - Cancel contracts here