Université Paris Cité

Python for Data Science — Final Project (v1)

Guillaume Wisniewski
guillaume.wisniewski@u-paris.fr

October 2024

- **Due date : 5 January 2026 at 8:00 am**
- **Remember that your code is most likely wrong if you need to write more than 5 lines to answer a question (including data loading). You can (normally) do the whole project using two for loops and without ever tabbing ; I managed to answer every question with a single instruction (at least for the first exercise).**
- **for each question, your report must include the line (or lines 😞) used to obtain the result, the result and, if necessary and only in this case, an explanation of your code.**
- **A (last) piece of advice : using the appropriate data type for each column will make your work much easier.**
- **also remember that commenting on your code and listening to your teachers' advice is a novice programmer's attitude, and you are no longer novices.**
- **each question will be graded with the following scale :**

grade	interpretation	percentage of points
A	your code matches the code I would have written	100 %
B	your code works but doesn't use pandas optimally / nicely	75 %
C	your code works but is completely twisted	50 %
D	your code doesn't answer the question	0 %

1 Analyzing `CommonVoice` *(20 pts)*

The aim of the first exercise is to analyse the `Common Voice` dataset [1]. This dataset contains sentences that have been recorded and transcribed by different speakers in various languages, and is used to train speech recognition systems among other tasks. Each speaker is potentially described by several metadata fields (e.g. gender, age, etc.). In this exercise, we will only use metadata, not the recordings themselves. The metadata can be downloaded from here. The goal of this exercise is to build test and training sets with specific characteristics.

1. *(1 pts)* For each language, determine the total duration of recordings and the number of unique speakers.
2. *(1 pts)* For each language and each gender, determine the total duration of recordings, the number of unique speakers, and the average, median, minimum and maximum number of recordings per speaker.
3. *(1 pts)* Which language has the highest proportion of recordings by speakers in the largest age group considered ?
4. *(1 pts)* Determine the number of different genders represented in the corpus. What percentage of recordings lack gender information ? Plot the gender distribution (including cases where gender information is missing). Do the same for age categories.
5. *(2 pts)* For each language and each gender, identify the 7 speakers who produced the highest number of recordings.
6. *(3 pts)* For each language, create a 1-hour test set and a training set as large as possible, ensuring that speakers appearing in the test set do not appear in the training set.
7. *(1 pts)* How many languages have more female than male recordings ?
8. *(3 pts)* For each language, create a training set containing 1 hour of recordings and a 10-minute test set such that : *i)* both are gender-balanced and *ii)* the speakers in the training and test sets are different.
9. *(2 pts)* Plot, for each language, a box plot representing the distribution of the number of words [2] per sentence.
10. *(3 pts)* For each language, find the 17 most frequent words after removing *stop words*.
11. *(2 pts)* Compute, for each language, the size of the available data both in terms of duration and in terms of number of words.

1. The official documentation of the dataset is available at `https://commonvoice.mozilla.org/en/datasets`.
2. You are expected to use proper tokenisation tools rather than a simple `split(" ")`.

2 Assessing the variability of `wav2vec2` representations *(10 pts)*

The aim of this exercise is to examine the inter- and intra-speaker variability of neural speech representations : are vector representations of the same word uttered twice by the same speaker closer to each other than the representations of the same word uttered by different speakers ? We will work with a corpus collected for a phonetic experiment. This corpus contains 31 sentences, each produced several times by 19 different speakers. The data are stored in a data frame. The "important" columns of this data frame are :

— `layer_i` : the representations at the i^{th} layer. This representation is stored as a `numpy array` of size number of frames $\times$ embedding size. In `wav2vec2`, the embedding dimension is 1,024 and each frame is sampled at 49 Hz (i.e., the i^{th} frame[3] corresponds to the signal between $\left(\frac{1}{49} \times (i-1)\right)$s and $\left(\frac{1}{49} \times i\right)$s).

— `words` : the word-level annotation. This is a list of `Interval` objects (from the `textgrid` library). Each object contains three pieces of information : the word spoken, its start time, and its end time.

The corpus can be downloaded from here (be careful, the file is quite large, $\simeq 1$ GB).

The code in Figure 1 aligns each word utterance with the vectors representing it (note that this code may contain bugs).

12. *(4 pts)* Explain each instruction of the `manatee` function.
13. *(1 pts)* Using the `manatee` function, create a data frame which, for each utterance of a word in the corpus, includes : the word, the speaker and their gender, and a matrix corresponding to the representation predicted by `wav2vec2` on the last layer. This matrix must have dimensions number of frames $\times$ 1,024.
14. *(4 pts)* Using the `librosa` library, compute the DTW distances between all pairs of utterances of the same word. To do so, for each word, generate all possible pairs of representations, compute the cosine distance between all frames of the two representations (i.e., if the representations are of size $n \times 1{,}024$ and $m \times 1{,}024$, the resulting *distance matrix* will be of size $n \times m$), and then compute the DTW on this distance matrix. Explain why we use DTW in this context.
15. *(1 pts)* Plot the distribution of distances for each word, distinguishing between cases where the two utterances were produced by the same speaker and cases where they were produced by different speakers. What do you conclude ? Figure 2 shows an example of the type of plot expected.

3. Assuming, somewhat naively, that the first frame is frame n°1.

```
def manatee(cow, yak, pelican, slug):
  cow = cow.to_frame().T.reset_index().copy()

  butterfly = cow[[yak]].explode(column=yak)
                        .apply(lambda x: {
                                  "mussel": x[yak].minTime,
                                  "end_time": x[yak].maxTime,
                                  "annotation": x[yak].mark,
                                },
                                axis=1,
                                result_type="expand",
                        )

  crane_fly = cow[[pelican]].explode(column=pelican)
  crane_fly["leaf_barnacle"] = slug
  crane_fly["leaf_barnacle"] = crane_fly["leaf_barnacle"].cumsum() - slug

  crane_fly["speaker"] = cow["speaker"]
  crane_fly["filename"] = cow["filename"]
  crane_fly["sentence"] = cow["sentence"]

  return pd.merge_asof(
        crane_fly, butterfly, left_on="leaf_barnacle", right_on="mussel"
  )

# we will assume that `df` contains the corpus and
# `layers` the list of layers that must be considered
for layer in layers:
  output = args.output_dir / layer
  output.mkdir(parents=True, exist_ok=True)

  for _, row in tqdm(df.iterrows(), total=df.shape[0]):
    d = align_representations(
              row,
              yak="word",
              pelican=layer,
              slug=1 / 49,
        )

    d["layer"] = layer
    d = d.rename(columns={layer: "repr"})
    d.to_pickle(output / f"{d.iloc[0]['filename'].stem}_aligned.pkl")

```

FIGURE 1 – Code to “help” you.

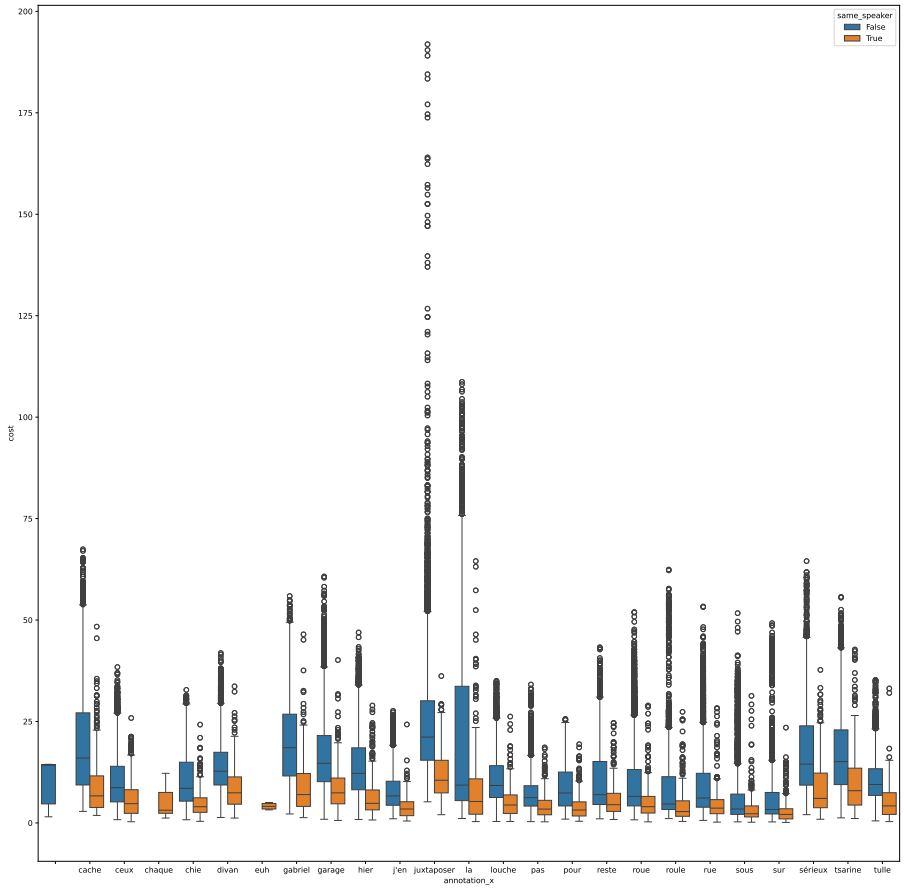

FIGURE 2 – Preview of the figure you (need to|want to|must) generate.