Adding new features to the Awkward-Array library

Maxym Naumchyk

While working with structures of data, the first idea that comes to your mind is that you can use Python lists or dictionaries. However, that takes a lot of computational resources and can take a lot of time if you’re working with a large dataset. Then, the thing you can use to quicken the data processing is NumPy. Instead of using for loops to apply a certain function to your data, it is using vectorization, which allows you to perform a function once over an entire dataset rather than individually to each element within it.

The next challenge you may come into, is when your data structure is irregular - meaning that the arrays in your dataframe have different shapes. For example, arrays like this:

\[
[[0, 1, 2], [], [3, 4], [5], [6, 7, 8, 9]]
\]

They are called ragged arrays. To store such data you could use Pandas library, but still, all your ragged arrays would need to have the same data types. So, when you deal with a ragged array that looks like this:

\[
[{{"x": 1}, "y": [2]}], [], [3.3, 4.4], [5.5], [6.6, 7.7, 8.8, 9.9]]
\]

That's where the Awkward-Array library comes in. It is using NumPy for fast computational possibilities and allows to use completely flexible data structures: be it irregular shape or array of records of different types.

My project would be to add new features to the Awkward library, among them:

● improving interconnection with similar features for ragged arrays like \textit{RaggedTensor} in TensorFlow’s library and \textit{NestedTensor} in PyTorch. Creating new functions to be able to convert awkward-array to/from Ragged and Nested Tensors without effort.
● adding custom QoL features like a function for finding a median, a function to get the first N highest/lowest entries, adding in-place operators support and a function to reduce an array to unique elements only, along a specific axis.
Also, I would write CI tests for the new implemented features. I will be using GitHub version control system and Windows Subsystem for Linux for writing new functions and testing them.

The timeline for the project would be:

- Getting familiar with *RaggedTensors* and *NestedTensors* (1 week)
- Implementing new to/from_raggedtensor functions (1 week)
- Implementing new to/from_nestedtensor functions (1 week)
- Adding a function for finding a median of an array (1 week)
- Adding a function to get the first N highest/lowest entries (1 week)
- Adding in-place operators support (1 week)
- Adding a function to reduce an array to unique elements only, along a specific axis (1 week)
- Finalizing the project (1 week)

Project will take place from late July until the end of September.