pymzavro Documentation

Release 0.5

Marius

August 15, 2015

CONTENTS

1 This is a short Tutorial of how to use pymzavro 3

2 mzMLWriter 5

3 reader 9

4 AvroSpectrumClass 11

5 Indices and tables 13

Python Module Index 15

Index 17

This is the documentation of pymzavro

Requirements: -None

Contents:

CHAPTER

ONE

THIS IS A SHORT TUTORIAL OF HOW TO USE PYMZAVRO

CHAPTER

TWO

MZMLWRITER

class mzMLWriter.**mzMLWriter**

Class to convert mzML to avro. Uses pyavroc and pymzML. It is possible that the size of a single record could cause problems with the default buffer size of pyavroc. Its recommened to patch the buffer size in datafile.c.

Example:

Import pymzavro and pymzml (to iterate)

```
>>> import pymzavro
>>> import pymzml
```

Open required files

```
>>> mzML = open("/home/marius/data/F/mzML/F04.mzML", "r")
>>> spectrumFile = open("F04_deflate.avro", "wb")
>>> spectrumSchema = open("spectrum.avsc","r")
>>> metaDataFile = open("F04meta_deflate.avro", "wb")
>>> typeDict = open("typeDict.json", "rb")
>>> metaDataSchema = open("fullSchema.avsc", "r")
>>> indexFile = "F04_Indexdeflate.json"
```

Create writer object and initialize it correctly

```
>>> writer = pymzavro.mzMLWriter.mzMLWriter()
```

```
>>> writer.init_file(mzML, spectrumFile, typeDict=typeDict, spectrumAvsc=spectrumSchema,
>>> writer.initmzAvroWriter()
```

Initialize reader (pymzML can be replaced by any other reader that offers the needed informations

```
>>> reader = pymzml.run.Reader("BSA3.avro")
```

Write metadata (optional), can be used for simple writer as well

```
>>> writer.writemzAvroMeta()
```

Iterator across the spectra and get original XML Tree and decoded m/z and intensity array, arrays are added to a dict (see schema maker):

```
>>> for spectrum in reader:
>>>     xmlSpectrum = spectrum.xmlTreeIterFree
>>>     mzArray = list(spectrum.mz)
>>>     iArray = list(spectrum.i)
>>>     dataDict = {"mzArray" : mzArray, "intensityArray" : iArray}
```

Write data to avro

```
>>>     writer.mzAvroWriter(xmlSpectrum, dataDict)
```

Write index

```
>>> writer.writeOffsetToJson()
```

advSeek(*index*)
This method is used to create an index based on the written blocks. To access a spectrum by index, seek() is used to jump to the start of the block and the an iterator goes across the spectra in that block to find the one with the given index.

param index The index of the spectrum

init_file(*mzML*, *avroFile*, *metaDataFile=None*, *typeDict=None*, *avro_schema=None*, *spectrumAvsc=None*, *indexJSON='index.json'*)
The init file method is used to provide information about the files the Writer should use. All files should be opened file like objects, except for the indexJSON.

param mzML The original mzML files

param avroFile The avro file the spectrumData is written to

param typeDict Needed to find data types

param avro_schema Schema of the whole mzML

param metaDataFile File the to which the metaData is written to

param spectrumAvsc Schema for writing the spectrum Data

param specialXML True if mzML is indexed, False if not

param indexJSON Filename of the index file

mzAvroWriter(*XML*, *writeDict*)
mzAvroWriter takes a spectrum XML as well as additional data stored in a dict that are appended to the file. Custom Datafields can be added to the schema using the appendCustomField method in the SchemaBuilder class. Usage:

```
>>> import pymzml
>>> import pymzavro
>>> mzAvroWriter = pymzavro.mzMLWriter.mzMLWriter()
>>> mzML = open("F04.mzML")
>>> spectrumFile = open("F04.avro", "wb")
>>> mzAvroWriter.init_file(mzML, spectrumFile)
>>> mzAvroWriter.initmzAvroWriter()
>>> mzMLReader = pymzml.run.Reader("F04.mzML")
>>> for spectrum in reader:
>>>     xml = spectrum.xmlTreeIterFree
>>>     mzAvroWriter.mzAvroWriter(xml, {"mzArray" : spectrum.mz, "intensityArray" : spectrum
```

Parameters

- **XML** – XML of one spectrum from spectrumlist
- **writeDict** – dict with additional data e.g. decoded data arrays

simplemzAvroWriter(*indexWrite='advanced'*)
simplemzAvroWriter is used to write spectrum data to an avro file, the information about the files are provided using init_file method. The indexWrite option is used to write an index based on the byte offsets of the fileto enable random access. The spectrum data is represented by a record that stores all the information that are stored in a spectrum child in the original mzML. Usage:

```
>>> import pymzavro
```

Needed files are opened

```
>>> mzML = open("/home/marius/data/F/mzML/F04.mzML", "r")
>>> spectrumFile = open("F04_deflate.avro", "wb")
>>> spectrumSchema = open("spectrum.avsc","r")
```

Initialize writer class

```
>>> writer = pymzavro.mzMLWriter.mzMLWriter()
>>> writer.init_file(mzML, spectrumFile, spectrumAvsc=spectrumSchema)
```

start simple conversion

```
>>> writer.simplemzAvroWriter()
```

param indexWrite Used to specify the indexing type that should be used, currently only advanced available

writeOffsetToJson()
Used to write the created index to a dict.

writemzAvroMeta()
Writes the metaData to the specified meta data avro file. Metadata are defined as all data from mzML that is not stored under a spectrum child.

CHAPTER

THREE

READER

class reader.**PymzAvroReader**(*avromz*, *metaDataFile=None*, *readerType=True*, *indexFile='index.json'*)

The pymzAvroReader class is used to give basic access to the stored file data. Its iterable and each iteration returns a avroSpectrum object that stores the information of the current spectrum and enables basic access to the data. The pymzAvroReader also enables random access to spectra using the rndSeek method. To initialize the reader, a spectrum file has to be handed over. Optionally a metadata file can be handed over. If a metadatafile is available, the metadata are added to spectrum Object. Example: #iterating across a whole avromz file and printing the mzArray of each spectrum

```
>>> import pymzavro
>>> spectrumAvroFile = open("file")
>>> run = pymzAvroReader(spectrumAvroFile)
>>> for spectrum in run:
>>>     mzArray = self.getmzArray()
>>>     print(mzArray)
```

#seek for a spectrum with a specific index

```
>>> import pymzavro
>>> spectrumAvroFile = open("file")
>>> indexFile = indexFilePath
>>> run = pymzAvroReader(spectrumAvroFile)
>>> run.rndSeek(index, indexFile)
```

More methods that provide provide data access please refer to the avroSpectrum class.

rndSeek(*index*, *avromz*)

Enables random seek :param index: number of the index :param avromz: path to the mzavroFile :return: dictionary representation of th current spectrum

CHAPTER

FOUR

AVROSPECTRUMCLASS

class avroSpectrum.**avroSpectrum**

Stores basic information about current spectra and and makes it accessible

addcvParamLocList(*addList*)

Used to add a list as a path to a cvParam to get cvParam data from there

:param addList: list with the path to the access target, currently only additional cvParams are enabled

getByAccession(*accession*)

Returns the value for the accession (obo Tag) from MSDict

param accession OBO Accession number, e.g.: "MS:1000014"

return value from cvParam for current accession

getIntensityArray()

Returns a list of intensity values of the current spectum, the values come from pymzML.spectrum.i, so if they were decoded in the original mzML file, they are decoded by mzML and then written to the avro file. :rtype: list :return: List of intensity values from the current spectrum

getMSDict()

Used to return a dict with all the cvParam informations extracted

rtype dict

return Returns the dict with all extracted cvParam informations

getSpectrum()

Returns a representation of the currently loaded spectrum, either object or dictionary, depending on datasource (iterating returns object, seeking returns dictionary) :return: self.spectrum

getmzArray()

Returns a list of mz values, thevalues come from pymzML.spectrum.mz, so if they were decoded in the original mzML file, they are decoded by mzML and then written to the avro file. :rtype: list :return: List of mzValues from the current spectrum

iterOvercvParam()

Reads cvParams specified in cvParamLocList and stores them with their accession number in a dictionary called MSDict

setData(*avroSpectrum*)

Loads the current spectrum data to the avroSpectrum class

param avroSpectrum Avro spectrum class

setFromDict(*currentDict*)
Function to build MSDict from a dictionary (e.g. for seeking) :param currentDict: Dict that represents the data of the spectrum :return:

setMSPropDict()

Sets minimum OBO list derived from pymzML minimum OBO, used to know wich attribute to get from cvParam

return

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

a

avroSpectrum, 11

m

mzMLWriter, 5

r

reader, 9

A

addcvParamLocList() (avroSpectrum.avroSpectrum method), 11
advSeek() (mzMLWriter.mzMLWriter method), 6
avroSpectrum (class in avroSpectrum), 11
avroSpectrum (module), 11

G

getByAccession() (avroSpectrum.avroSpectrum method), 11
getIntensityArray() (avroSpectrum.avroSpectrum method), 11
getMSDict() (avroSpectrum.avroSpectrum method), 11
getmzArray() (avroSpectrum.avroSpectrum method), 11
getSpectrum() (avroSpectrum.avroSpectrum method), 11

I

init_file() (mzMLWriter.mzMLWriter method), 6
iterOvercvParam() (avroSpectrum.avroSpectrum method), 11

M

mzAvroWriter() (mzMLWriter.mzMLWriter method), 6
mzMLWriter (class in mzMLWriter), 5
mzMLWriter (module), 5

P

PymzAvroReader (class in reader), 9

R

reader (module), 9
rndSeek() (reader.PymzAvroReader method), 9

S

setData() (avroSpectrum.avroSpectrum method), 11
setFromDict() (avroSpectrum.avroSpectrum method), 11
setMSPropDict() (avroSpectrum.avroSpectrum method), 12
simplemzAvroWriter() (mzMLWriter.mzMLWriter method), 6

W

writemzAvroMeta() (mzMLWriter.mzMLWriter method), 7
writeOffsetToJson() (mzMLWriter.mzMLWriter method), 7