1. Method
   1. Work on this project was performed under Dr. Rafael Gomez-Bombarelli of the MIT Department of Materials Science and Engineering. Files are available on GitHub at <https://github.com/DarnellGranberry/MStoGraphNN>. Neural network architecture was based on Wengong Jin’s Junction Tree Variational Autoencoder (cf. <https://github.com/wengong-jin/icml18-jtnn> and <https://arxiv.org/pdf/1802.04364.pdf>).
   2. Overview
      1. Acquire and prepare MS data
      2. Process array with LSTM then supply output to jtnn decoder’s forward method
   3. Acquisition of mass spectrometry data
      1. Organic compound MS data is available as text files from the Japanese Mass Spec Society at <https://github.com/MassBank/MassBank-data>.
         1. N.B.: update will reportedly be coming soon (~Sept 2018) to correct decimal point errors that were generated from converting data CD-ROM into online database format.
      2. Files were compiled and converted to a list of dictionaries, each containing the compound’s name, given and canonical SMILES and InChI strings, m/z array (as a numpy array), and details on the spectroscopy like instrument and ion mode.
         1. Some samples did not have SMILES, so their InChI strings were used with the cirpy python package to fetch the SMILES strings.
      3. The m/z arrays were converted to pandas DataFrames, and a pandas histogram was used to visualize and bin the number and range of peaks for each sample. To cut down the size of the input matrices, the arrays were restricted to 100 peaks, containing ~93% of the data here.
      4. The arrays that had fewer than 100 peak entries were padded to length with zeroes (just prior to model). This may decrease the accuracy of the peak distribution. This might be remedied by a common scale for all samples; this is not feasible for this dataset, because of the great variance in the methods of measurement (positive vs negative ion MS, different voltage levels, etc.).
      5. Most of the above actions were performed by running “MassSpec\_DataParser.py”.
      6. The records are serialized in “MS\_Records.pickle” (file too large to upload to GitHub).
   4. Neural network implementation
      1. One epoch took approximately 6-7 hours on Dell XPS 13 CPU (Intel 8th gen, i7).
      2. Current hyperparameters: LSTM input size=2, batch size =40, hidden size = latent size = 2, output size = 256 (currently unused), epochs = 1, learning rate = 0.001.
      3. Hidden size and latent size must currently be the same; square matrices result in the fewest errors in implementation.
      4. The model checks the shape of the input tensor in numerous places and pads with zeroes to ensure operations like concatenation, stacking, and matrix multiplication occur without error.
      5. In “MStoGraphNN.py” there exists an “issue\_smiles” list object containing 282 samples that resulted in errors when passed as input for the VAE. The source of some of these errors was a failure to remove the carriage return symbol from the cluster vocabulary that has since been solved. Others were due to a size mismatch error in a matrix multiplication, and a lack of tensors containing bond information in the decoding layer. Solutions to these issues are yet to be found.
2. ­Needed Corrections/Future Work
   1. Saving each layer or defining another model to include all layers—including jtnn decoder layer. Layers are not defined/contained in a single class, so they may each need their own unique state dictionary
   2. Determine whether data can be moved to a common scale, or if all peaks should remain at the beginning.
   3. Experiment with different numbers of LSTM units.
   4. Experiment with different 1-dimensional convolutions.
   5. Optimize hyperparameters, perhaps via SigOpt.