**ML 23/24-10 Multi-Sequence Learning with language semantic**

Multisequence Learning is the process of learning many sequences like:

ABCD, DDDE, CDEE, FFGE, …

This can be used in industrial solutions to solve many problems. For example, the sequence ABCDDEE can be an array of music notes, and HTM can be used to recognize songs.

The sample is implemented in the source-code file MultisequenceLearning.cs. The method RunMultiSequenceLearningExperiment demonstrates how sequences are learned and then predicted. This sample is fully implemented and works fine.

Your first task is to analyze the existing code and understand how learning sequences and prediction work. Then implement a new method RunLanguageSemanticExperiment, that improves the existing RunMultiSequenceLearningExperiment. The new method should automatically read learning sequences from a file and learn them. With this modification, the existing experiment sample can be run. However, you must have enough data in the file (long text), which will be split into two groups. 90% of the data will be used for training and 10% for testing. Model will be trained with the training data (90% sample). After learning is completed, the sample should read testing subsequences (10% sample) from another file and calculate the prediction accuracy.

The file should contain a very long sequence, which contains the text of some book. In the beginning you might use a few pages of some book or even some song. You must try different sources. For example, assume there is the following text in the file.

You know the day destroys the night  
Night divides the day  
Tried to run, tried to hide  
  
Break on through to the other side  
Break on through to the other side  
Break on through to the other side, yeah  
  
We chased our pleasures here  
Dug our treasures there  
A-can you still recall the time we cried?  
  
Break on through to the other side

. . .

The sequence is the array of characters like You know the day. You should load the file and create the sequence from characters, where every character is represented by its ASCII code. Following code illustrates how to create the sequence from text:

string text = "You know the day . . .”;

int[] S1 = new int[text.Length];

for(int n = 0; n< text.Length;n++)

{

S1[n] = text[n];

Console.Write($"{S1[n]} ");

}

The sequence will contain following elements:

89 111 117 32 107 110 111 119 32 116 104 101 32 100 97 121

In the real experiment, you should remove all control characters like \r, \n, \t etc.

Your task is to find a way to calculate the accuracy. One possibility is to calculate the binary cross-entropy depending on the length of the dataset. The expected bits in the SDR (currently active cells produced by TM) are compared with the predicted probabilities. Your code must start unattended, load the text file, train the model, and output the accuracy as a loos function as a binary cross-entropy. You will run the model on different sizes of the training file and compare the result.

Finally, you have to implement predicting (inferring) code, which enables the user to enter some text and the model should continue to generate the text after the given text. For example, the user enters:

We chased

The model continues with:

We chased our pleasures here Dug our treasures there

You also have to make it possible that different sequences are created, not only sequences hitting the character with the maximal probability.

We chased something else here

The predicting code represents the completion engine as used by GPT.