Methods :

We use the model training corpus from the Ding et al. study, consisting of 60, 4 word (monosyllabic) English sentences, of which we use 47 sentences having the NP-VP structure. The five experimental conditions are recreated using these sentences. That is - grammatical sentences, concatenated NP and VP sequences, word salad condition consisting of randomly ordered words, and random patterns of the kind – V, V, N, Adj. We run tests on three model varieties – A custom model trained with the paradigm explained below, the Ding stimuli presented to the Berkeley Neural Parser as four words presented per second (without any blanks), the same stimuli presented to the pretrained BERT. We analyse the hidden layer activations of these models

Training data :

We reduce the sampling frequency from 200Hz to 20Hz, and in the first step represent the training data as a T1 = 47 x 20 x N matrix, where every word is represented in five time steps. The first of these time steps is a one hot encoding signifying a blank condition. The Frank and Yang model considers word-embedding appearance at approximately 50 ms after onset. In a similar manner, in this training setup, the input vector representation of the word appears from the 2nd until the 5th timesteps. We use two ways to represent a word – First, from the Google word2vec pretrained model, we obtain N(=300) dimensional vectors, and second, as simple one-hot encodings. The output labels consist of the input time steps shifted ahead in time. Therefore, a typical output label consists of the following sequence – w1,w1,w1,w1,ENDW,w2...,ENDW,... ...,w4,w4,w4,w4,ENDS. Where we represent the end of a word with a one hot encoding of ‘ENDW’, and end of sentence with a different one hot encoding of ‘ENDS’.

Using T1, we up-sample the permutations of the 47 sentences to create T2 = 20 x (20 \* 13) x N dimensional matrix, treating the first dimension as number of samples, second dimension as time steps, and third dimension as vector length (260 time steps for every sample and 20 samples). In order to ensure that T2 does not teach the model to treat every fourth word as completing a sentence, we create a staggered version of T2. That is, we sample 240 continuous time steps in T2, each starting at l = [0,19). In the 240 timesteps, in all those sequences which do not have the 4 full words, we replace the ‘ENDS’ signal with ‘ENDW’.

Custom Model :

We use a Recurrent Neural Network model containing a single input layer, a hidden layer consisting of 100 LSTM (Long Short Term Memory) units, connected to a Densely connected output layer. We use the Adam Optimizer, and train the model until it plateaus for 10 epochs, and updates at 100 steps per epoch. We train four different custom models for two conditions – Presence of ‘ENDS’ signal or only ‘ENDW’ signal, and using the pretrained vectors as input representation or only one hot encodings.

Results :

We simulate the trained model with N(=200) samples, each having T(=240) time steps, and get the hidden state activations for all units H(=100), in the form of a matrix of shape NxTxH. For every sample, and every unit, we perform the Fourier transform to get the frequency response of shape – T/2 x H (one response per unit), which is then averaged across all units. This response is averaged across all samples. (No windowing for now, looks good without)

BERT and Berkeley Neural Parser :

We simulated the pretrained BERT and pretrained Berkeley Neural Parser [...] with data a matrix D (= 20 x (13\*4)), where 20 samples contain 13, 4 word sequences. Interesting (not very strong results in 1 Hz, strong in 2 Hz. See plots-bert-new-datagen) results in case of BERT. The Neural parser shows stronger results (see plots-parser-new-datagen)!