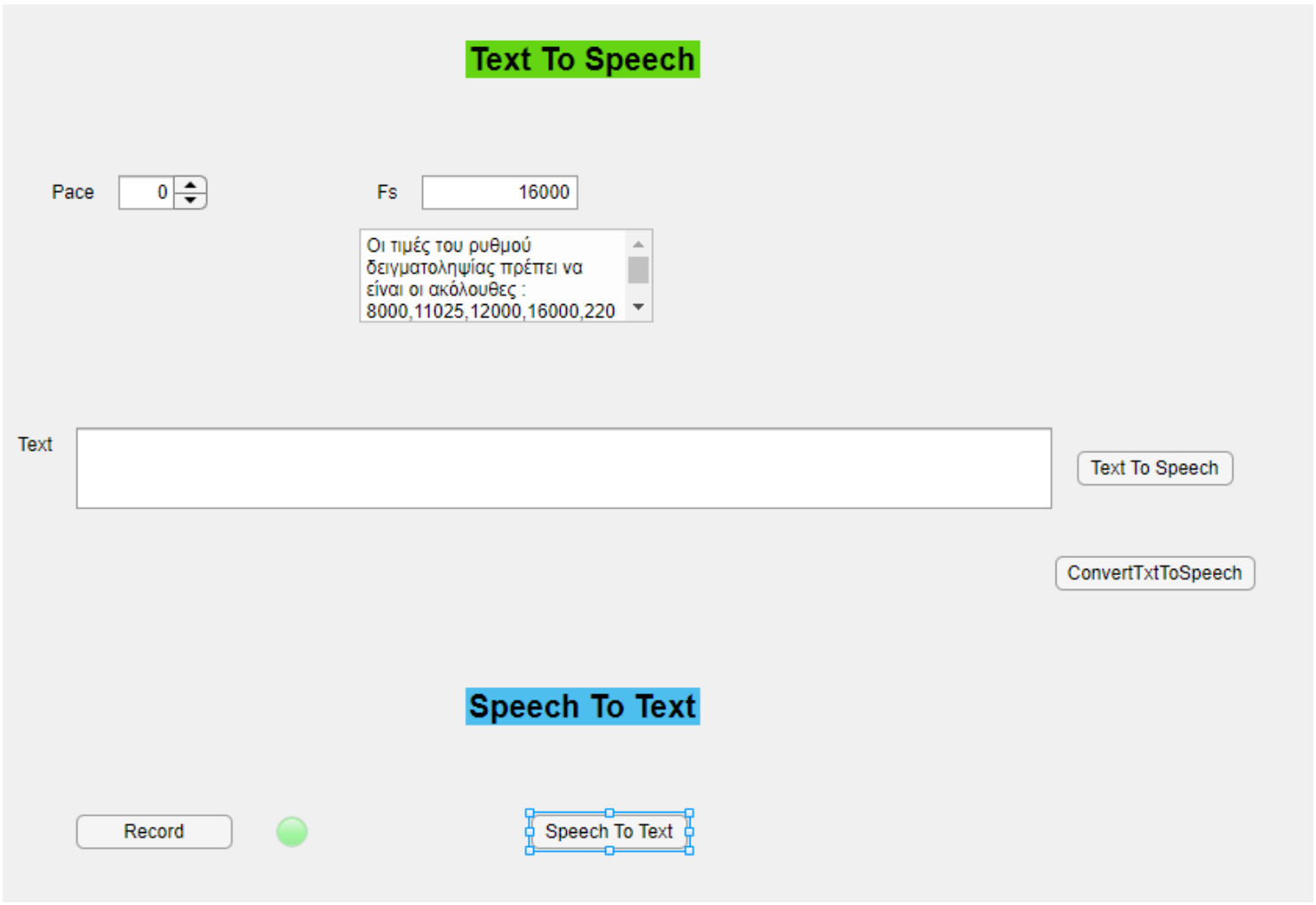
Αλέξανδρος Βυθούλκας 2013021
Απόστολος Καμπλιώνης 2015121

Επεξεργασία Ομιλίας και Ήχου - Απαλλακτική Εργασία:
"Μετατροπή του γραπτού λόγου σε ομιλία (Text To Speech – Speech To Text) και αναγνώριση γλώσσας"

Checkpoint - Παραδοτέο 2 / Παρουσίαση προόδου εργασίας(1/4/22)

Όσο αφορά το 2ο Παραδοτέο, ερχόμενοι από την ολοκληρωμένη υλοποίηση της λειτουργίας **Text-To-Speech**(TTS), εστιάσαμε στη σύνταξη κώδικα για την αντίστροφη λειτουργία **Speech-To-Text**(STT) καθώς και το αντίστοιχο GUI Button αυτής.

```
% Button pushed function: SpeechToTextButton
function SpeechToTextButtonPushed(app, event)
    file = uigetfile();
    txt = wav2vec(audioread(file),app.FsEditField.Value);
    %txt = char(x);
    msgbox('The transformation to text is ready','Speech To Text');
    answer = questdlg('Do you want to save the text file ?','Save File');
       if answer == 'Yes', writematrix(txt,'sttfile.txt'); end %save('sttfile.txt', 'txt','-ascii'); end

end
```



Υλοποίηση του STT κομματιού με την δημιουργία του κατάλληλου UI

Mathworks link : [wav2vec-2.0 - File Exchange - MATLAB Central (mathworks.com)]

Συγκεκριμένα δημιουργήθηκαν 2 buttons. Το πρώτο εκτελεί την λειτουργία **Record** του ηχητικού input(με τη χρήση **Record** GUI Button μέσω App Designer) και εμφάνιση αντίστοιχων dialog boxes για την ενημέρωση του χρήστη ως προς το recording state(start/stop) καθώς και ερώτημα για την αποθήκευση ή μη της εισόδου σε αρχείο **wav**. Το δεύτερο button εστιάζει στη μετατροπή του παραπάνω αποθηκευμένου ηχητικού σε κείμενο και αποθήκευση αυτού σε **txt** αρχείο. Πραγματοποιείται εφαρμογή του μοντέλου **wav2vec,** το οποίο εκπαιδεύτηκε χρησιμοποιώντας το LibriSpeech dataset και αποτελείται από 4 στάδια:

```
function txt = wav2vec(X,fs)
%WAV2VEC Speech-to-text transcription
% txt = wav2vec(x,fs) performs speech-to-text transcription on the audio
% signal x with sample rate fs. The output txt is a string. The function
% uses the pretrained baseline 960 hrs system with the corresponding
% wave2vec 2.0 architecture.
persistent featureEncoder Learnables Dictionary
if isempty(Learnables) || isempty(featureEncoder) || isempty(Dictionary)
    if ~exist("wav2vec2-base-960.mat",'file')
        error("wav2vec:ModelNotFound","Unable to access the pretrained system.\nMake sure the required files are
    end
    load("wav2vec2-base-960.mat","Learnables","Dictionary")
    if canUseGPU()
        Learnables = dlupdate(@(x)gpuArray(x),Learnables);
    end
    featureEncoder = constructFeatureEncoder(Learnables.featureEncoder);
end

% Normalize and resample if necessary
X = wav2vecPreprocess(X,fs);

% Feature encoding
X = squeeze(stripdims(featureEncoder.forward(X)));

% Positional encoding
X = positionalEncoder(X,Learnables.positionalEncoder);

% Transformer encoding
X = transformerEncoder(X,Learnables.transformerEncoder);

% Text decoding
```

```
% Text decoding
txt = textDecoder(X,Dictionary);

end
```

- Το πρώτο ονομάζεται **Feature Encoder** όπου το ηχητικό αρχείο εισόδου διέρχεται μέσα από 7 σύστροφα μπλόκς μιας διάστασης. Το πρώτο μπλοκ περιέχει ένα επίπεδο κανονικοποίησης μεταξύ του σύστροφου μπλοκ και της συνάρτησης Gaussian Error Linear Units (GELU). Η έξοδος των σύστροφων μπλόκς διέρχεται πάλι από ένα στρώμα κανονικοποίησης πριν περάσουν από το γραμμικό επίπεδο.

Το συνολικό περιεχόμενο από την περιοχή λήψης του κωδικοποιητή είναι 400 δείγματα τα οποία συμβαδίζουν με 25 ms στα 16 kHz δειγματοληψίας fs.

```
function net = constructFeatureEncoder(params)
%CONSTRUCTFEATUREENCODER Construct feature encoder
% net = constructFeatureEncoder(params) returns a dlnetwork consisting of 7
% 1-D convoluational blocks for feature encoding.
layers = [
    sequenceInputLayer(1,MinLength=400,Name="input")

    convolution1dLayer(10,512,Stride=5,Weights=params.conv1d_1.Weights,Name="conv1d_1")
    instanceNormalizationLayer(Scale=params.instancenorm_1.Scale,Offset=params.instancenorm_1.Offset,Name="instancenorm_1")
    customLayer.geluLayer(Name="gelu_1")

    convolution1dLayer(3,512,Stride=2,Weights=params.conv1d_2.Weights,Name="conv1d_2")
    customLayer.geluLayer(Name="gelu_2")

    convolution1dLayer(3,512,Stride=2,Weights=params.conv1d_3.Weights,Name="conv1d_3")
    customLayer.geluLayer(Name="gelu_3")

    convolution1dLayer(3,512,Stride=2,Weights=params.conv1d_4.Weights,Name="conv1d_4")
    customLayer.geluLayer(Name="gelu_4")

    convolution1dLayer(3,512,Stride=2,Weights=params.conv1d_5.Weights,Name="conv1d_5")
    customLayer.geluLayer(Name="gelu_5")

    convolution1dLayer(2,512,Stride=2,Weights=params.conv1d_6.Weights,Name="conv1d_6")
    customLayer.geluLayer(Name="gelu_6")

    convolution1dLayer(2,512,Stride=2,Weights=params.conv1d_7.Weights,Name="conv1d_7")
    customLayer.geluLayer(Name="gelu_7")

    layerNormalizationLayer(Offset=params.layernorm_1.Offset,Scale=params.layernorm_1.Scale,Name="layernorm_1")
```

```
    fullyConnectedLayer(768,Weights=params.fc_1.Weights,Bias=params.fc_1.Bias,Name="fc_1")
    ];

b = layerGraph(layers);
net = dlnetwork(b);
end
```

- Το δεύτερο λέγεται **Positional Encoding**. Σε αυτό, τα λανθάνοντα χαρακτηριστικά που προέκυψαν από το προηγούμενο βήμα εισέρχονται σε ένα σύστροφο γκρουπ μιας διάστασης για να δημιουργηθεί ένα σχετικό διάνυσμα θέσεων το οποίο θα προστεθεί στα λανθάνοντα χαρακτηριστικά για την κωδικοποίηση της θέσης των συγγενικών χαρακτηριστικών μεταξύ τους.

```
function X = positionalEncoder(X,params)
%POSITIONALENCODER Positional encoding
% dlY = positionalEncoder(Y,params) encodes Y with a relative
% positional encoding vector then applies layer normalization.

% 1-D convolution layer
Xe = positionalEmbedding(X,params.positionalEmbedding);

% Residual connection and normalization
X = Xe + X;
X = layernorm(X,params.layernorm.offset,params.layernorm.scaleFactor,DataFormat="CB");

    function z = positionalEmbedding(x,params)
        %POSITIONALEMBEDDING Positional embedding
        % z = positionalEmbedding(x,params) performs grouped 1-d
        % convolution and returns a relative positional embedding vector
        % described in [1].

        % 1-D convolution
        z = dlconv(x,params.Weights,params.Bias,DataFormat="CTB",WeightsFormat="TUCU",Stride=1,Padding=64);
        z = z(:,1:end-1);

        % GELU activation
        z = gelu(z);
    end
end
```

- Το τρίτο στάδιο λέγεται **Contex Network** και αποτελείται από 12 block κωδικοποιητών. Κάθε μπλοκ εφαρμόζει μια multi-head προσέγγιση και τροφοδοτεί το επόμενο διαδοχικό μπλοκ. Το επόμενο διαδοχικό μπλοκ αποτελείται από 2 πλήρως συνδεδεμένα επίπεδα κανονικοποίησης χωριζόμενα από ένα GELU layer. Στην multi-head προσέγγιση κάθε μπλοκ του context network έχει 3 γραμμικά (fully-connected) layers των οποίων οι έξοδοι είναι το Q (query), K (key), και V (value) διανύσματα. Τα διανύσματα Q, K, and V στην συνέχεια χωρίζονται σε 12 (ο αριθμός των κεφαλών μέσα στο σύστημα) μη επικαλυπτόμενων sections. Η Scaled dot-product προσέγγιση εφαρμόζεται σε κάθε χώρισμα ξεχωριστά και τα αποτελέσματα είναι συνδεδεμένα. Η έξοδος διέρχεται μέσω ενός γραμμικού επιπέδου για να σχηματιστεί η τελική multi-head προσέγγιση.

```
function Y = transformerEncoder(X,params)
%TRANSFORMERENCODER Transformer encoding
% Y = transformerEncoder(X,params) passes X through 12 encoder blocks
% in sequence then applies a fully-connected layer.

X = encoderBlock(X,params.block_1);
X = encoderBlock(X,params.block_2);
X = encoderBlock(X,params.block_3);
X = encoderBlock(X,params.block_4);
X = encoderBlock(X,params.block_5);
X = encoderBlock(X,params.block_6);
X = encoderBlock(X,params.block_7);
X = encoderBlock(X,params.block_8);
X = encoderBlock(X,params.block_9);
X = encoderBlock(X,params.block_10);
X = encoderBlock(X,params.block_11);
X = encoderBlock(X,params.block_12);

Y = fullyconnect(X,params.fc.Weights,params.fc.Bias,DataFormat="CB");

    function X = encoderBlock(X,params)
        %ENCODERBLOCK Encoder block
        % X = encoderBlock(X,params) passes Y through an encoder
        % block.
        %
        % See section 3.1 of [1].

        % Multi-Head Attention
        xa = multiheadAttention(X',params.multiheadAttention);

        % Add & Norm
        X = X + xa;
        X = layernorm(X,params.layernorm_1.offset,params.layernorm_1.scaleFactor,DataFormat="CB");

        % Feed Forward
        xff = fullyconnect(X,params.fc_1.Weights,params.fc_1.Bias,DataFormat="CB");
        xff = gelu(xff);
        xff = fullyconnect(xff,params.fc_2.Weights,params.fc_2.Bias,DataFormat="CB");

        % Add & Norm
        X = X + xff;
        X = layernorm(X,params.layernorm_2.offset,params.layernorm_2.scaleFactor,DataFormat="CB");

    end
end
```

```
function A = multiheadAttention(X,params)
%MULTIHEADATTENTION Multi-head attention
% A = multiheadAttention(X,params) performs multi-head attention
% described in section 3.2.2 of [1].

numHeads = 12;

% Linear
K = fullyconnect(X,params.fc_1.Weights,params.fc_1.Bias,DataFormat="BC");
V = fullyconnect(X,params.fc_2.Weights,params.fc_2.Bias,DataFormat="BC");
Q = fullyconnect(X,params.fc_3.Weights,params.fc_3.Bias,DataFormat="BC");

% Scaled dot-product attention
A = scaledDotProductAttention(Q,K,V,numHeads);

% Concatenation
A = iMergeHeads(A);

% Linear
A = fullyconnect(A,params.fc_4.Weights,params.fc_4.Bias,DataFormat="CB");

    function A = scaledDotProductAttention(Q,K,V,numHeads)
        %SCALEDDOTPRODUCTATTENTION Scaled dot-product attention
        % A = scaledDotProductAttention(Q,K,V,numHeads)
        % performs scaled dot-product attention described in section 3.2.1 of [1].

        % The function is vectorized to apply attention to all heads
        % simultaneously.

        numFeatures = size(K,1);
```

```
        numFeatures = size(K,1);

        % Matrix multiplication
        K = iSplitHeadsQ(K,numFeatures,numHeads);
        Q = iSplitHeads(Q,numFeatures,numHeads);
        W = dlmtimes(K,Q);

        % Scale
        W = W./sqrt(size(Q,1));

        % Softmax
        W = softmax(W,DataFormat="CTUB");

        % Matrix multiplication
        V = iSplitHeads(V,numFeatures,numHeads);
        A = dlmtimes(V,W);

        function Z = iSplitHeads(X,numFeatures,numHeads)
            % For the key and value vectors, permute to put the dimension for
            % the heads last and the sequence length second. This enables
            % batched matrix multiplication to compute attention for all of the
            % heads at once.
            X = reshape(X,numFeatures/numHeads,numHeads,[],1);
            Z = permute(X,[1,3,2]);
        end
        function Z = iSplitHeadsQ(X,numFeatures,numHeads)
            % For the query vector, permute to put the dimension of the heads
            % last and the sequence length first. This enables batched matrix
            % multiplication to compute attention for all of the heads at once.
            X = reshape(X,numFeatures/numHeads,numHeads,[],1);
            Z = permute(X,[3,1,2]);
```

```
        function Z = iSplitHeadsQ(X,numFeatures,numHeads)
            % For the query vector, permute to put the dimension of the heads
            % last and the sequence length first. This enables batched matrix
            % multiplication to compute attention for all of the heads at once.
            X = reshape(X,numFeatures/numHeads,numHeads,[],1);
            Z = permute(X,[3,1,2]);
        end
    end
    function Z = iMergeHeads(X)
        % Merge the numFeatures-by-sequenceLength-by-numHeads array to a
        % (numFeatures*numHeads)-by-sequenceLength array
        X = permute(X,[1,3,2]);
        Z = reshape(X,size(X,1)*size(X,2),[]);
    end

end
```

- Το τέταρτο και τελευταίο στάδιο ονομάζεται **Text Decoding** και όπως λέει και το όνομα εδώ γίνεται αποκωδικοποίηση στην τελική μορφή του κειμένου. Το μοντέλο είναι εκπαιδευμένο χρησιμοποιώντας connectionist temporal classification (CTC).

```
function txt = textDecoder(Y,dict)
%TEXTDECODER Text decoding
% txt = textDecoder(y,dict) decodes wav2vec 2.0 output y to text txt
% using the map in dict.

txt = '';
for ii = 1:size(Y,2)
    [~,argmax] = max(Y(:,ii));
    if ~endsWith(txt,dict{argmax})
        txt = txt + dict{argmax};
    end
end

txt = replace(txt,'<pad>','');
txt = replace(txt,'|',' ');
txt = char(txt);
if ~isempty(txt) && txt(end)==' '
    txt = txt(1:end-1);
end
txt = string(txt);

end
```

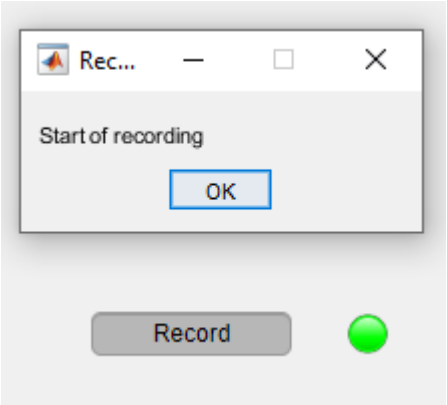
Gaussian Error Linear Unit (GELU) Activation

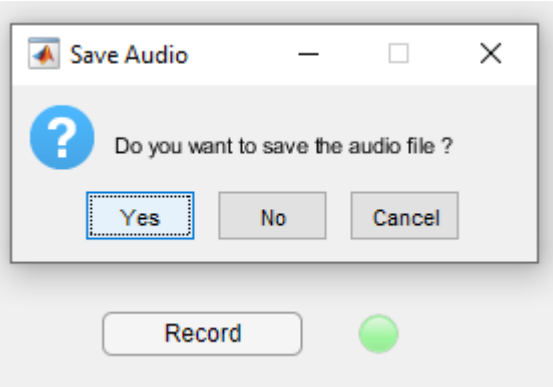
```
function Z = gelu(x)
%GELU Apply GELU activation
% Z = gelu(x) applies Gaussian error linear unit (GELU) activation to the
% input x.

Z = 0.5*x.*(1 + tanh(sqrt(2/pi)*(x + 0.044715*(x.^3))));

end
```

Με την ολοκλήρωση της ηχογράφησης(unpressed button state), εμφανίζεται το αντίστοιχο dialog box που προτρέπει τον χρήστη να αποθηκεύσει το ηχητικό input ως αρχείο wav.
Με το button **Speech To Text** ο χρήστης έχει τη δυνατότητα να μετατρέψει το περιεχόμενο του σήματος από το wav αρχείο σε μορφή κειμένου txt.

Παραθέτουμε screenshots από την μέχρι τώρα πρόοδο

```
function RecordButtonValueChanged(app, event)
    value = app.RecordButton.Value; % h timh toy state button true / false
    %app.currently_recording = true;
    datatype = 'uint8'; %o typos me ton opoio 8a apo8hkeytoyn ta dedomena
    nbits = 16; %o ari8mos twn bit poy 8a parei ws eisodo h synarthshs audiorecorder
    ch = 1; % gia na to hxografhmeno akoysthko kyma na einai mono kai oxi stereo
    fs = app.FsEditField.Value; %h timh poy fs apo to interface
    if value %elegxos an h timh toy state einai true
        app.Lamp.Enable = 'on'; % energopoiei thn lampa
        f = msgbox('Start of recording','Recording'); % dialogbox gia thn enarxh
        app.recorder = audiorecorder(fs,nbits,ch); % h dhmioyrgia enos antikeimenoy audiorecc
        record(app.recorder); % enarxh ths leitoyrgias hxografishs
    else
        stop(app.recorder); %paysh ths hxografishs
        app.Lamp.Enable = 'off'; % apenergopoiei thn lampa
        x = getaudiodata(app.recorder,datatype);% apo8hkeyei topika to apotelesma ths hxograf
        %app.currently_recording = false;
        answer = questdlg('Do you want to save the audio file ?','Save Audio');
        % dhmioyrgia enos dialog box poy rwtaei ton xrhsth an 8elei
        %na apo8hkeysei ston diskoto hxhtiko arxeio poy exei dhmioyrgh8ei

            if answer == 'Yes', audiowrite('samplerec.wav',x,fs); end
            %ean h apanthsh einai Yes tote apouhkeyse to hxhtiko
            %wav ston disko
    end
```


Rec...
Start of recording
OK
Record


Save Audio
Do you want to save the audio file ?
Yes
No
Cancel
Record