ELEC-E5510 – Speech Recognition

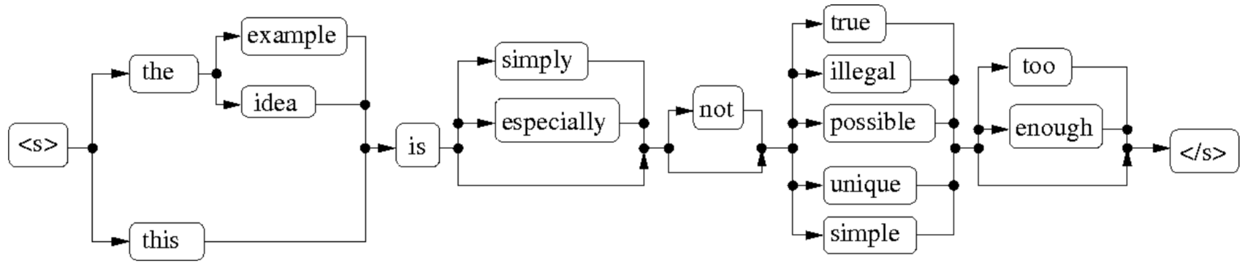
Assignment 4

Nguyen Xuan Binh, 887799

22/11/2023

Question 1

(a) The picture below defines an artificial grammar for simple statement sentences. Define a corresponding grammar in HTK format and convert it to a recognition network using HParse



Include in the report the grammar definition you constructed, the commands you used, and the individual recognition results as reported by HResults: HResults -h -t -I $data/grammar.mlf /dev/null grammar.rec
Why did the recognizer make mistakes?

First, I created a grammar.txt that describes the grammar from the provided diagram. The grammar rule is written in the HTK format, and there are four vocabulary groups:

the (example | idea)
is (simply | especially)
not (true | illegal | possible | unique| simple)
(too | enough) </s>

SR_ex4 > grammar.txt

```
$noun = example | idea;
$adverb = simply | especially;
$adj = true | illegal | possible | unique | simple;
$adv = too | enough;

( \<s\> ( ( the $noun | this) is [ $adverb ] [not] $adj [$adv] ) \<\/s\> )
```

Next, I converted this grammar.txt to a recognition network by this command:

$ HParse grammar.txt grammar_net.htk

Where grammar net.htk is the output file containing the recognition network. We can use this
$ HSGen grammar_net.htk $data/grammar.vocab

to generate some sentences permitted by these grammar rules. We can see that all generated sentences follows the diagram above

```
nguyenb5@kosh ~/SR_ex4 % HSGen grammar_net.htk $data/grammar.vocab
<s> the example is true </s>
<s> this is simple enough </s>
<s> this is especially simple enough </s>
<s> the idea is unique enough </s>
<s> the example is especially unique enough </s>
<s> the idea is illegal </s>
<s> the example is not true enough </s>
<s> the idea is not simple too </s>
<s> this is unique too </s>
<s> this is illegal too </s>
<s> this is possible enough </s>
```

Using the recognition network and the above mentioned models, recognize a small test set $data/grammar.scp. We use HVite for decoding by this command
**$ HVite -T 1 -i grammar.rec -H $data/macros -H $data/hmmdefs **
**-C $data/config -w grammar_net.htk -s 10.0 -t 200.0 **
-S $data/grammar.scp $data/grammar.dict $data/tiedlist

```
nguyenb5@kosh ~/SR_ex4 % HVite -T 1 -i grammar.rec -H $data/macros -H $data/hmmdefs \
-C $data/config -w grammar_net.htk -s 10.0 -t 200.0 \
-S $data/grammar.scp $data/grammar.dict $data/tiedlist
Read 9798 physical / 62402 logical HMMs
Read lattice with 23 nodes / 42 arcs
Created network with 525 nodes / 765 links
File: /work/courses/T/S/89/5150/general/ex4/grammardata/gs01.mfcc
<s> the example is not unique </s>  ==  [240 frames] -68.0090 [Ac=-16322.2 LM=0.0] (Act=44.0)
File: /work/courses/T/S/89/5150/general/ex4/grammardata/gs02.mfcc
<s> the idea is simple enough </s>  ==  [261 frames] -65.9332 [Ac=-17208.6 LM=0.0] (Act=40.4)
File: /work/courses/T/S/89/5150/general/ex4/grammardata/gs03.mfcc
<s> this is simply not true </s>  ==  [184 frames] -65.4109 [Ac=-12035.6 LM=0.0] (Act=45.8)
File: /work/courses/T/S/89/5150/general/ex4/grammardata/gs04.mfcc
<s> this is illegal too </s>  ==  [256 frames] -63.3889 [Ac=-16227.5 LM=0.0] (Act=25.6)
File: /work/courses/T/S/89/5150/general/ex4/grammardata/gs05.mfcc
<s> this is possible </s>  ==  [278 frames] -71.4944 [Ac=-19875.4 LM=0.0] (Act=68.8)
File: /work/courses/T/S/89/5150/general/ex4/grammardata/gs06.mfcc
<s> this is especially true </s>  ==  [300 frames] -71.2049 [Ac=-21361.5 LM=0.0] (Act=81.0)
File: /work/courses/T/S/89/5150/general/ex4/grammardata/gs07.mfcc
<s> this is simple enough </s>  ==  [373 frames] -73.9560 [Ac=-27585.6 LM=0.0] (Act=38.2)
```

The reported results by HResults is

$ HResults -h -t -I $data/grammar.mlf /dev/null grammar.rec

```
nguyenb5@kosh ~/SR_ex4 % HResults -h -t -I $data/grammar.mlf /dev/null grammar.rec
Aligned transcription: /work/courses/T/S/89/5150/general/ex4/grammardata/gs07.lab vs
grammardata/gs07.rec
 LAB: <s> this example is not    unique </s>
 REC: <s> this         is simple enough </s>
    ,-------------------------------------------------------------.
    | HTK Results Analysis at Sun Nov 19 11:41:24 2023            |
    | Ref: /work/courses/T/S/89/5150/general/ex4/grammar.mlf      |
    | Rec: grammar.rec                                            |
    |=============================================================|
    |           # Snt |  Corr    Sub    Del    Ins    Err  S. Err |
    |-------------------------------------------------------------|
    | Sum/Avg |    7  |  93.33   4.44   2.22   0.00   6.67  14.29 |
    `-------------------------------------------------------------'
```

The recognizer makes an Word Error Rate (WER) of 6.67% based on the table

To know why this recognizer makes mistake, we can look at the reported example
LAB: <s> this example is not unique </s>
REC: <s> this is simple enough </s>

It is clear that the test sentence (LAB) does not follow the grammar rule provided by the diagram, which is encoded in grammar.txt. In this definition, the beginning of the sentence can only either be “this” or “the $noun”. However, the recognized sentence (REC) starts with “this $noun”, which is not included in the grammar rules

(b) Let's try the same recognition task with an n-gram language model. File $data/grammar.sent includes training sentences generated from the grammar. These were generated using HTK tool HSGen. Using SRI tool ngram-count as instructed in Exercise 3, train two 2-gram models, with and without smoothing (for the smoothed model, use options -interpolate -cdiscount1 0 -cdiscount2 0.5).
You need to modify the unsmoothed language model in a text editor (it is text-based ARPA format), because ngram-count compensates round-off errors with non-infinite back-off weights: Under \1-grams: section, replace all back-off weights (the third column) with -99, which represents minus infinite. This makes sure the language model can not generate unseen word sequences.
Build a recognition network out of the language models with HBuild. For example,
HBuild -s "<s>" "</s>" -n grammar_2gram.lm $data/grammar.vocab grammar_2gram_net.htk
creates a recognition network from 2-gram model grammar_2gram.lm. Run the recognition and see the results with HResults as in part a). Include the commands and the output of HResults to the report. How did the recognition results change with these language models? Explain why.

UNSMOOTHED LANGUAGE MODEL

At first, I trained the unsmoothed 2-gram language model with the command

$ ngram-count -order 2 -text $data/grammar.sent -lm unsmoothed.lm

The language model is stored in the file named unsmoothed.lm. To prevent the generation of word sequences not present in the training data, the value in the third column of the \1-grams section needs to be modified to -99. This adjustment ensures that the model does not assign probabilities to unseen word sequences.

```
SR_ex4 > unsmoothed.lm
\1-grams:
-0.8129134  </s>
-99 <s> -99
-1.290035   enough  -99
-1.290035   especially  -7.668898
-1.290035   example -99
-1.290035   idea    -99
-1.511883   illegal -7.553777
-0.8129134  is  -7.652635
-1.113943   not -7.375425
-1.511883   possible    -7.553777
-1.511883   simple  -7.553777
-1.290035   simply  -7.668898
-0.9890046  the -7.434815
-1.290035   this    -99
-1.290035   too -99
-1.511883   true    -7.553777
-1.511883   unique  -7.553777
```

Original output language model

```
SR_ex4 > unsmoothed.lm
\1-grams:
-0.8129134  </s>
-99 <s> -99
-1.290035   enough  -99
-1.290035   especially  -99
-1.290035   example -99
-1.290035   idea    -99
-1.511883   illegal -99
-0.8129134  is  -99
-1.113943   not -99
-1.511883   possible    -99
-1.511883   simple  -99
-1.290035   simply  -99
-0.9890046  the -99
-1.290035   this    -99
-1.290035   too -99
-1.511883   true    -99
-1.511883   unique  -99
```

Modified language model

Next, I built the recognition network from the LM model unsmoothed.lm and the network is generated as the file grammar_2gram_unsmoothed_net.htk

$ HBuild -s "<s>" "</s>" -n unsmoothed.lm $data/grammar.vocab grammar_2gram_unsmoothed_net.htk

I used the Hvite command to test the unsmoothed recognizer on the grammar.scp. The results are saved in the file grammer_unsmoothed.rec

**$ HVite -T 1 -i grammar_unsmoothed.rec -H $data/macros -H $data/hmmdefs **
**-C $data/config -w grammar_net.htk -s 10.0 -t 200.0 **
-S $data/grammar.scp $data/grammar.dict $data/tiedlist

Finally, the recognition results are reported for the unsmoothed model

$ HResults -h -t -I $data/grammar.mlf /dev/null grammar_unsmoothed.rec

```
nguyenb5@kosh ~/SR_ex4 % HResults -h -t -I $data/grammar.mlf /dev/null grammar_unsmoothed.rec
Aligned transcription: /work/courses/T/S/89/5150/general/ex4/grammardata/gs07.lab vs /work/cou
grammardata/gs07.rec
 LAB: <s> this example is not    unique </s>
 REC: <s> this         is simple enough </s>
    ,-------------------------------------------------------------.
    | HTK Results Analysis at Sun Nov 19 13:05:44 2023             |
    | Ref: /work/courses/T/S/89/5150/general/ex4/grammar.mlf       |
    | Rec: grammar_unsmoothed.rec                                  |
    |=============================================================|
    |           # Snt |  Corr    Sub    Del    Ins    Err  S. Err |
    |-------------------------------------------------------------|
    | Sum/Avg |    7  |  93.33   4.44   2.22   0.00   6.67  14.29 |
    `-------------------------------------------------------------'
```

There is no change in the WER compared to Question 1a, which is still 6.67%

SMOOTHED LANGUAGE MODEL

At first, I trained the smoothed 2-gram language model with the command

**$ ngram-count -order 2 -interpolate -cdiscount1 0 -cdiscount2 0.5 **
-text $data/grammar.sent -lm smoothed.lm

Now there is no need to manually change the language model like the unsmoothed version. I then built a recognition network for the smoothed LM with HBuild.

$ HBuild -s "<s>" "</s>" -n smoothed.lm $data/grammar.vocab
grammar_2gram_smoothed_net.htk

I used the Hvite command to test the smoothed recognizer on the grammar.scp. The results are saved in the file grammer_smoothed.rec

**$ HVite -T 1 -i grammar_smoothed.rec -H $data/macros -H $data/hmmdefs **
**-C $data/config -w grammar_2gram_smoothed_net.htk -s 10.0 -t 200.0 **
-S $data/grammar.scp $data/grammar.dict $data/tiedlist

```
nguyenb5@kosh ~/SR_ex4 % HVite -T 1 -i grammar_smoothed.rec -H $data/macros -H $data/hmmdefs \
-C $data/config -w grammar_2gram_smoothed_net.htk -s 10.0 -t 200.0 \
-S $data/grammar.scp $data/grammar.dict $data/tiedlist

Read 9798 physical / 62402 logical HMMs
Read lattice with 18 nodes / 81 arcs
Created network with 1228 nodes / 2578 links
File: /work/courses/T/S/89/5150/general/ex4/grammardata/gs01.mfcc
<s> the example is not unique </s>  ==  [240 frames] -68.2444 [Ac=-16322.2 LM=-56.5] (Act=195.8)
File: /work/courses/T/S/89/5150/general/ex4/grammardata/gs02.mfcc
<s> the idea is simple enough </s>  ==  [261 frames] -66.1504 [Ac=-17208.6 LM=-56.7] (Act=171.7)
File: /work/courses/T/S/89/5150/general/ex4/grammardata/gs03.mfcc
<s> this is simply not true </s>  ==  [184 frames] -65.7175 [Ac=-12035.6 LM=-56.4] (Act=177.1)
File: /work/courses/T/S/89/5150/general/ex4/grammardata/gs04.mfcc
<s> this is illegal too </s>  ==  [256 frames] -63.6099 [Ac=-16227.5 LM=-56.6] (Act=161.1)
File: /work/courses/T/S/89/5150/general/ex4/grammardata/gs05.mfcc
<s> this is possible </s>  ==  [278 frames] -71.6977 [Ac=-19875.4 LM=-56.5] (Act=417.1)
File: /work/courses/T/S/89/5150/general/ex4/grammardata/gs06.mfcc
<s> this is especially true </s>  ==  [300 frames] -71.3935 [Ac=-21361.5 LM=-56.6] (Act=397.9)
File: /work/courses/T/S/89/5150/general/ex4/grammardata/gs07.mfcc
<s> this example is not unique </s>  ==  [373 frames] -68.0895 [Ac=-25259.4 LM=-138.0] (Act=225.2)
```

Finally, the recognition results are reported for the smoothed model

$ HResults -h -t -I $data/grammar.mlf /dev/null grammar_smoothed.rec

```
nguyenb5@kosh ~/SR_ex4 % HResults -h -t -I $data/grammar.mlf /dev/null grammar_smoothed.rec
    ,-------------------------------------------------------------.
    | HTK Results Analysis at Sun Nov 19 13:21:23 2023            |
    | Ref: /work/courses/T/S/89/5150/general/ex4/grammar.mlf      |
    | Rec: grammar_smoothed.rec                                   |
    |=============================================================|
    |           # Snt |  Corr    Sub    Del    Ins    Err  S. Err |
    |-------------------------------------------------------------|
    | Sum/Avg |    7  | 100.00   0.00   0.00   0.00   0.00   0.00 |
    `-------------------------------------------------------------'
```

Now it is different, as the accuracy of the smoothed LM manages to reach 100% (0% WER)

Reason why the smoothed model achieves perfect accuracy: Smoothing techniques distribute a portion of the probability to n-grams that were not observed during training (allow non-trivial probabilities), allowing the model to account for possible but unseen word combinations. As a result, a smoothed model can handle sequences that deviate from the strict grammatical structures found in the training data, leading to improved performance and potentially reducing the error rate to zero. In contrast, an unsmoothed model assigns a probability of zero to any n-gram not present in the training set. Consequently, both the unsmoothed model and the recognizer from part a) yield the same Word Error Rate (WER) because they handle unseen n-grams in the same way.

Question 2

a) Using the 3-gram language model, recognize the WSJ evaluation set with language model weights 12.0, 14.0, 16.0 and 18.0. Use beam pruning threshold 200.0. Report the commands you used and the word error rate for each of the language model weight (you can omit -t switch in HResults to have less output). Which language model weight gave the best recognition results?

The evaluation set from the Wall Street Journal (WSJ) is processed using language model weights of 12, 14, 16, and 18, applying a beam pruning threshold set to 200. I use a for loop to avoid repeating the commands

$ for i in {12.0,14.0,16.0,18.0};
**do HDdecode -T 1 -C $data/config -C $data/config.hdecode -S $data/wsj_5k_eval.scp **
**-i results_2a_$i.mlf -H $data/macros -H $data/hmmdefs -t 200.0 -s $i **
-w $data/wsj_5k.3gram.lm $data/wsj_5k.hdecode.dict $data/tiedlist ; done

Then, we can compute the WER for each generated LM using a loop

$ for i in {12.0,14.0,16.0,18.0}; do HResults -h -I $data/wsj_5k_eval.mlf /dev/null results_2a_$i.mlf; done

The WER for each model are reported below
w = 12, WER = 5.83
w = 14, WER = 5.00
w = 16, WER = 5.42
w = 18, WER = 6.04

```
,-------------------------------------------------------------.
| HTK Results Analysis at Sun Nov 19 16:05:10 2023            |
| Ref: /work/courses/T/S/89/5150/general/ex4/wsj_5k_eval.mlf  |
| Rec: results12.0.mlf                                        |
|=============================================================|
|          # Snt |  Corr    Sub    Del    Ins    Err  S. Err  |
|-------------------------------------------------------------|
| Sum/Avg |   54 |  95.83   3.96   0.21   1.67   5.83  46.30  |
`-------------------------------------------------------------'
```

Weight -s = 12.0, Pruning threshold -t = 200

```
,----------------------------------------------------------------------.
| HTK Results Analysis at Sun Nov 19 16:05:10 2023                     |
| Ref: /work/courses/T/S/89/5150/general/ex4/wsj_5k_eval.mlf           |
| Rec: results14.0.mlf                                                 |
|======================================================================|
|            # Snt |  Corr    Sub    Del    Ins    Err  S. Err        |
|----------------------------------------------------------------------|
| Sum/Avg |   54   |  96.46   3.33   0.21   1.46   5.00  44.44        |
`----------------------------------------------------------------------'
```

Weight -s = 14.0, Pruning threshold -t = 200

```
,----------------------------------------------------------------------.
| HTK Results Analysis at Sun Nov 19 16:05:10 2023                     |
| Ref: /work/courses/T/S/89/5150/general/ex4/wsj_5k_eval.mlf           |
| Rec: results16.0.mlf                                                 |
|======================================================================|
|            # Snt |  Corr    Sub    Del    Ins    Err  S. Err        |
|----------------------------------------------------------------------|
| Sum/Avg |   54   |  95.94   3.75   0.31   1.35   5.42  48.15        |
`----------------------------------------------------------------------'
```

Weight -s = 16.0, Pruning threshold -t = 200

```
,----------------------------------------------------------------------.
| HTK Results Analysis at Sun Nov 19 16:05:10 2023                     |
| Ref: /work/courses/T/S/89/5150/general/ex4/wsj_5k_eval.mlf           |
| Rec: results18.0.mlf                                                 |
|======================================================================|
|            # Snt |  Corr    Sub    Del    Ins    Err  S. Err        |
|----------------------------------------------------------------------|
| Sum/Avg |   54   |  95.73   3.96   0.31   1.77   6.04  50.00        |
`----------------------------------------------------------------------'
```

Weight -s = 18.0, Pruning threshold -t = 200

Therefore, the language model with weight 16 has the best recognition results because it has lowest WER at 5.00%

b) Run the recognition again with language model weights 12.0 and 18.0, but now with beam pruning threshold 220.0. Compare the WERs to the results of the a) part. Why does the larger language model weight require a larger beam threshold as well?

In this part, I only changed the value of the threshold flag to -t 220 and removed 14 and 16 from the loop. The procedure is still similar to Question 2a

**$ for i in {12.0,18.0}; do HDecode -T 1 -C $data/config -C $data/config.hdecode -S $data/wsj_5k_eval.scp **
**-i results_2b_$i.mlf -H $data/macros -H $data/hmmdefs -t 220.0 -s $i **
-w $data/wsj_5k.3gram.lm $data/wsj_5k.hdecode.dict $data/tiedlist ; done

Then, we can compute the WER for each generated LM using a loop

$ for i in {12.0,18.0}; do
HResults -h -I $data/wsj_5k_eval.mlf /dev/null results_2b_$i.mlf; done

The WER for each model are reported below
w = 12, WER = 5.83
w = 18, WER = 5.31

```
,-------------------------------------------------------------.
| HTK Results Analysis at Sun Nov 19 16:45:43 2023            |
| Ref: /work/courses/T/S/89/5150/general/ex4/wsj_5k_eval.mlf  |
| Rec: results_2b_12.0.mlf                                    |
|=============================================================|
|           # Snt |  Corr    Sub    Del    Ins    Err  S. Err |
|-------------------------------------------------------------|
| Sum/Avg |   54  |  95.62   4.17   0.21   1.46   5.83  44.44 |
`-------------------------------------------------------------'
```

Weight -s = 12.0, Pruning threshold -t = 220

```
,-------------------------------------------------------------.
| HTK Results Analysis at Sun Nov 19 16:45:43 2023            |
| Ref: /work/courses/T/S/89/5150/general/ex4/wsj_5k_eval.mlf  |
| Rec: results_2b_18.0.mlf                                    |
|=============================================================|
|           # Snt |  Corr    Sub    Del    Ins    Err  S. Err |
|-------------------------------------------------------------|
| Sum/Avg |   54  |  96.04   3.65   0.31   1.35   5.31  46.30 |
`-------------------------------------------------------------'
```

Weight -s = 18.0, Pruning threshold -t = 220

Observation: the WER for weight 12 does not change when we change the threshold from t=200 to t=220. However, the WER for weight 18 decreases from 6.04 to 5.31 when we change the threshold from t=200 to t=220

Reason: As the language model weight increases, the speech recognition system increasingly prefers frequently occurring sentences, enhancing the influence of the language model. This results in a greater number of infrequent sentence hypotheses that must be pruned to maintain computational efficiency. Consequently, raising the beam threshold becomes necessary to eliminate these less common hypotheses, which increases with higher language model weights.

Question 3

Do the following two tasks without running HDecode again. Report the commands used and the WER results from HResults.

a) Extract the recognition results from the original lattices (BEFORE rescoring) and evaluate their WER. Use language model weight 18.

I extracted the recognition results from the original lattices (BEFORE rescoring) and evaluate their WER. Use language model weight 18. with lattice-tool and HResults commands

**$ lattice-tool -htk-lmscale 18 -in-lattice-list original_lattices.list \
-read-htk -viterbi-decode | $data/viterbi2mlf.pl > lattices/rec_w18_original.mlf**

$ HResults -h -I $data/wsj_5k_eval.mlf /dev/null lattices/rec_w18_original.mlf

The WER for the model is 5.94

```
,-------------------------------------------------------------.
| HTK Results Analysis at Sun Nov 19 17:09:28 2023            |
| Ref: /work/courses/T/S/89/5150/general/ex4/wsj_5k_eval.mlf  |
| Rec: lattices/rec_w18_original.mlf                          |
|=============================================================|
|          # Snt |  Corr    Sub    Del    Ins    Err  S. Err  |
|-------------------------------------------------------------|
| Sum/Avg |   54  |  94.90   4.69   0.42   0.83   5.94  46.30 |
`-------------------------------------------------------------'
```

WER of model weight 18

b) Fetch the best paths from the 4-gram rescored lattices using language model weights 10.0, 14.0, 18.0, 22.0, 26.0 and 30.0. Report the WER results. Which weight now gave the best result?

I fetched the best paths from the 4-gram rescored lattices using language model weights 10.0, 14.0, 18.0, 22.0, 26.0 and 30.0 and reported the WER results

```
$ for i in {10,14,18,22,26,30};
do lattice-tool -htk-lmscale $i -in-lattice-list rescored_lattices.list \
-read-htk -viterbi-decode | $data/viterbi2mlf.pl > rescored/rec_w$i.mlf; done
```

```
$ for i in {10,14,18,22,26,30}; do HResults -h -I $data/wsj_5k_eval.mlf \
/dev/null rescored/rec_w$i.mlf; done
```

The WER for each model are reported below
w = 10, WER = 6.04   w = 22, WER = 4.27
w = 14, WER = 5.42   w = 26, WER = 5.00
w = 18, WER = 5.00   w = 30, WER = 5.00

```
,-------------------------------------------------------------------.
| HTK Results Analysis at Sun Nov 19 17:18:19 2023                  |
| Ref: /work/courses/T/S/89/5150/general/ex4/wsj_5k_eval.mlf        |
| Rec: rescored/rec_w10.mlf                                         |
|===================================================================|
|          # Snt |  Corr    Sub    Del    Ins    Err  S. Err        |
|-------------------------------------------------------------------|
| Sum/Avg |  54  | 95.83   3.96   0.21   1.88   6.04  48.15         |
`-------------------------------------------------------------------'
```

WER for weight 10 model

```
,-------------------------------------------------------------------.
| HTK Results Analysis at Sun Nov 19 17:18:19 2023                  |
| Ref: /work/courses/T/S/89/5150/general/ex4/wsj_5k_eval.mlf        |
| Rec: rescored/rec_w14.mlf                                         |
|===================================================================|
|          # Snt |  Corr    Sub    Del    Ins    Err  S. Err        |
|-------------------------------------------------------------------|
| Sum/Avg |  54  | 96.04   3.65   0.31   1.46   5.42  42.59         |
`-------------------------------------------------------------------'
```

WER for weight 14 model

```
,-------------------------------------------------------------.
| HTK Results Analysis at Sun Nov 19 17:18:19 2023             |
| Ref: /work/courses/T/S/89/5150/general/ex4/wsj_5k_eval.mlf   |
| Rec: rescored/rec_w18.mlf                                    |
|=============================================================|
|           # Snt |  Corr    Sub    Del    Ins    Err  S. Err |
|-------------------------------------------------------------|
| Sum/Avg |   54  |  96.25   3.44   0.31   1.25   5.00  42.59 |
`-------------------------------------------------------------'
```

WER for weight 18 model

```
,-------------------------------------------------------------.
| HTK Results Analysis at Sun Nov 19 17:18:19 2023             |
| Ref: /work/courses/T/S/89/5150/general/ex4/wsj_5k_eval.mlf   |
| Rec: rescored/rec_w22.mlf                                    |
|=============================================================|
|           # Snt |  Corr    Sub    Del    Ins    Err  S. Err |
|-------------------------------------------------------------|
| Sum/Avg |   54  |  96.67   3.12   0.21   0.94   4.27  40.74 |
`-------------------------------------------------------------'
```

WER for weight 22 model

```
,-------------------------------------------------------------.
| HTK Results Analysis at Sun Nov 19 17:18:19 2023             |
| Ref: /work/courses/T/S/89/5150/general/ex4/wsj_5k_eval.mlf   |
| Rec: rescored/rec_w26.mlf                                    |
|=============================================================|
|           # Snt |  Corr    Sub    Del    Ins    Err  S. Err |
|-------------------------------------------------------------|
| Sum/Avg |   54  |  96.15   3.44   0.42   1.15   5.00  42.59 |
`-------------------------------------------------------------'
```

WER for weight 26 model

```
,---------------------------------------------------------------
| HTK Results Analysis at Sun Nov 19 17:18:19 2023
| Ref: /work/courses/T/S/89/5150/general/ex4/wsj_5k_eval.mlf
| Rec: rescored/rec_w30.mlf
|===============================================================
|           # Snt |  Corr    Sub    Del    Ins    Err  S. Err
|---------------------------------------------------------------
| Sum/Avg |   54  |  95.83   3.65   0.52   0.83   5.00  40.74
`---------------------------------------------------------------
```

WER for weight 30 model

Therefore, the language model with weight 22 has the best recognition results because it has lowest WER at 4.27%

Question 4

Consider different speech recognition applications. When would you use an n-gram model trained from a large text corpus? When would you use other kinds of language models?

When would you use an n-gram model trained from a large text corpus?
N-gram models are most effective when trained on a large and representative text corpus, as they rely heavily on the frequency of word sequences in the training data. They are suitable for applications with extensive vocabularies and require accurate predictions based on common usage, such as in grammar correction or language translation.

When would you use other kinds of language models?
N-gram models struggle with out-of-vocabulary (OOV) words and require smoothing techniques to handle them, which may still be insufficient if the training corpus is too small or doesn't match the test domain well. For such scenarios or when the application involves unpredictable contexts, alternative language models like the Connectionist Temporal Classification (CTC) or neural networks (NNs) are preferred, as they are less dependent on the exact word frequencies and can generalize better to unseen data due to their context dependence collocations.