Assignment 4

Gengcong Yan - 1009903
ELEC-E5510 - Speech Recognition

December 1, 2021

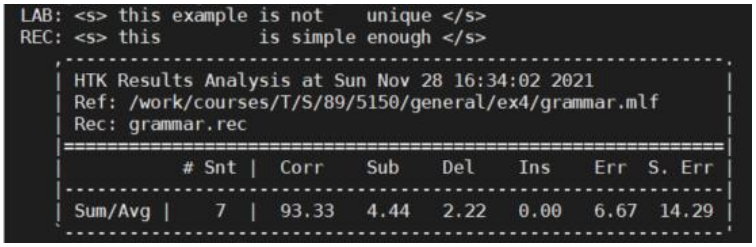

Figure 1: Recognition Result

1 Question 1

1.1 A

The grammar definitions I constructed are shown below:

```
$Noun= example| idea;
$Adv = simply| especially;
$Adj = true|illegal|possible|unique|simple;
$AdvLast= too|enough;

( \<s\>  ( the $Noun | this) is [$Adv] [not] $Adj [$AdvLast]
   \<\/s\> )
```

The commands used in question are below:

```
#commands

HParse gram.txt grammar_net.htk

HSGen grammar_net.htk $data/grammar.vocab

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

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

The recognition results shows in Fig 1. Err rate is 6.67%. we can also see different kinds of error, substitutions, deletions, and insertions. They represent the operations that need to be modified to get the correct result.

I think the reason why the recognition still make mistakes is that the sentence length is variable in the data. That is, when making predictions, it is sometimes difficult to determine whether a word or multiple words are represented in phonemes, resulting in deletion and substitution errors in prediction. There is no silence between words, we do not know transition time from word to word, making the prediction harder.

Model	Sub	Del	Ins	Err
2-gram	4.44	2.22	0.00	6.67
2-gram(Smooth)	0.00	0.00	0.00	0.00

Table 1: Recognition results

1.2 B

```
#commands

#without smoothing
ngram-count -order 2  -text $data/grammar.sent -lm 2gram.lm

HBuild -s "<s>" "</s>" -n 2gram.lm $data/grammar.vocab 2
   gram_net.htk

HVite -T 1 -i 2grammar.rec -H $data/macros -H $data/hmmdefs \
  -C $data/config -w 2gram_net.htk -s 10.0 -t 200.0 \
  -S $data/grammar.scp $data/grammar.dict $data/tiedlist

HResults -h -t -I $data/grammar.mlf /dev/null 2grammar.rec


#Smoothing
ngram-count -order 2 -interpolate -cdiscount1 0 -cdiscount2 0.5
    -text $data/grammar.sent -lm smooth2gram.lm

HBuild -s "<s>" "</s>" -n smooth2gram.lm $data/grammar.vocab
   smooth2gram_net.htk

HVite -T 1 -i smooth2grammar.rec -H $data/macros -H $data/
   hmmdefs \
  -C $data/config -w smooth2gram_net.htk -s 10.0 -t 200.0 \
  -S $data/grammar.scp $data/grammar.dict $data/tiedlist

HResults -h -t -I $data/grammar.mlf /dev/null smooth2grammar.
   rec
```

The 2-gram model without smoothing have the same performance with the model in 1.A. But the 2-gram model with smoothing predict the small evaluation test perfectly. Smoothing can avoid the appearance of extreme values in the probability calculation, giving a little tendency to the less likely words sequences, thus making the overall result better.

Model	Sub	Del	Ins	Err
W12B200	3.96	0.21	1.67	5.83
W14B200	3.33	0.21	1.46	**5.00**
W16B200	3.75	0.31	1.35	5.42
W18B200	3.96	0.31	1.77	6.04
W12B220	4.17	0.21	1.46	5.83
W18B220	3.65	0.31	1.35	**5.31**

Table 2: Recognition results with different parameters

2 Question 2

2.1 A

The commands in this part are basically the same except language model weights and beam pruning threshold. So **only 1 group commands shows below.**

```
# Commands
HDecode -T 1 -C $data/config -C $data/config.hdecode -S $data/
   wsj_5k_eval.scp \
  -i results_12_200.mlf -H $data/macros -H $data/hmmdefs -t
     200.0 -s 12.0 \
  -w $data/wsj_5k.3gram.lm $data/wsj_5k.hdecode.dict $data/
     tiedlist

HResults -h -t -I $data/wsj_5k_eval.mlf /dev/null
   results_12_200.mlf
```

The recognition results shows in Table 2. The language model with weight=14 gives the best recognition results.

2.2 B

The recognition results also shows in Table 2. When we increase the value of beam pruning threshold, the prediction results of the models are improved, compared to the original models. As can be seen in the introduction, we know that the larger the weight value, the more the recognition favors common sentences defined by the language model. This will cause more path duplication, so we use larger beam values, remove very unlikely hypothesis, and terminate unnecessary repeated computations, not only to speed up the process, but also to increase the accuracy.

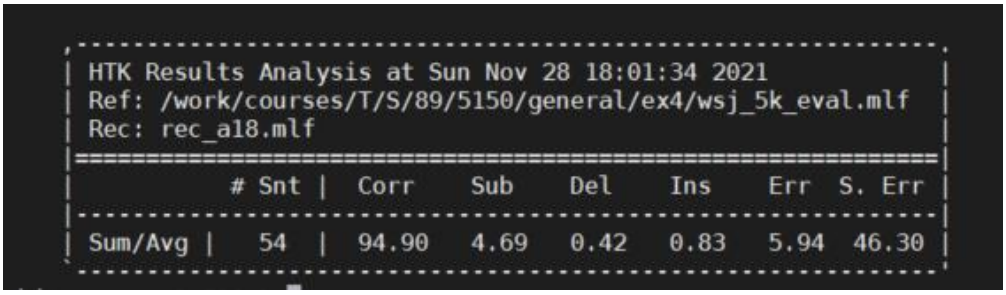

Figure 2: Recognition Result

3 Question 3

3.1 A

```
#Commands

mkdir lattices

HDecode -T 1 -C $data/config -C $data/config.hdecode -S $data/
   wsj_5k_eval.scp \
  -H $data/macros -H $data/hmmdefs -z htk -l lattices -t 175.0
     -s 10.0 \
  -w $data/wsj_5k.2gram.lm $data/wsj_5k.hdecode.dict $data/
     tiedlist

ls lattices/*.htk.gz > original_lattices.list

lattice-tool -htk-lmscale 18 -in-lattice-list original_lattices
   .list \
  -read-htk -viterbi-decode | $data/viterbi2mlf.pl > rec_a18.
     mlf

HResults -h -I $data/wsj_5k_eval.mlf /dev/null rec_a18.mlf
```

The recognition result from he original lattices shows in Fig 2. The WER is 5.94%.

Model	Sub	Del	Ins	Err
W10	3.96	0.21	1.88	6.04
W14	3.65	0.31	1.46	5.42
W18	3.44	0.31	1.25	5.00
W22	3.12	0.21	0.94	**4.27**
W26	3.44	0.42	1.15	5.00
W30	3.65	0.52	0.83	5.00

Table 3: Recognition results from the 4-gram rescored lattices

3.2 B

```
#Commands

ls lattices/*.htk.gz > original_lattices.list

lattice-tool -order 4 -in-lattice-list original_lattices.list \
  -read-htk -lm $data/wsj_5k.4gram.lm.gz -write-htk -out-
     lattice-dir rescored

ls rescored/*.htk.gz > rescored_lattices.list

# change weights in 10 14 18 22 26 30
lattice-tool -htk-lmscale 10 -in-lattice-list rescored_lattices
   .list \
  -read-htk -viterbi-decode | $data/viterbi2mlf.pl > rescored/
     rec10.mlf

HResults -h -I $data/wsj_5k_eval.mlf /dev/null rescored/rec10.
   mlf
```

The recognition results shows in Table 3. The best performance is given by the language model using weight=22.

4 Question 4

When speech recognition system only needs specific features, such as recognizing numbers for dialing like in this assignment, there are only a few fixed words as dictionary. In this case the grammar based recognition network may work better. Doing specific grammatical generalization for specific scenario limits the recognition range, thus achieving better results. N-grams model can be applied into more complex recognition tasks with huge vocabularies or also require prediction beyond the vocabularies. Of course, we also need to know that when n is too large, the entire phrase space becomes too sparse and the probability of many words phrases is close to 0. And the increasing size of the entire model is also a problem. In lecture 5 of End-to-End ASR, we know that HMM-based systems tends to do better with

limited resources. But End-to-End system can more easily run on mobile devices. It directly map speech to text in applications.