Report

STRUCTURE	STATES	ARCS	FCPW
BASELINE	116	230	109421.5
SILENCE(UNIGRAM)	176	350	160839.8
SILENCE(BIGRAM)	196	460	160839.8

Table 1. Different structures in this coursework.

1. Introduction and Initial systems

1.1. Structure

This assignment involves a total of three different WFST structural designs, without considering the probability. Table 1 summarizes these three structures. The first structure, *baseline*, is used as the initial system architecture, and the latter two structures will be mentioned in subsequent experiments. Figure 1 shows this *baseline* structure. For the *baseline* experiment, the weights are not set and are the default initial values. Adjustments to various weights are not considered in this section, which will be covered in the next set of experiments in next section.

1.2. Dateset

The voice data set provided by this assignment is 180 segments of English audio recorded voluntarily by students of the 2019-2020 UoE ASR course [1]. Each sample is composed of speech and transcription pairs, and all words come from the ten words including a, of, peck, peppers, peter, picked, pickled, piper, the and where's. Since this is not an authoritative data set, flaws in the data set itself will affect the overall experiment. This fact also appears in this assignment, which will be discussed in detail in a later section.

1.3. Code Base

Some of the basic functions in the code used in the experiments in this report are derived from the lab solutions of the 2019-2020 UoE ASR course [2]. For better coding style, WFST-related functions are integrated into a class, which can be viewed in the code. Although the lab solution has some authority, it has not been tested extensively, and there may be hidden risks. At the time of writing this report, I discovered a problem caused by a problem with the underlying code, which will be described in detail in a later section.

[1] https://homepages.inf.ed.ac.uk/s1680221/

[2] https://www.inf.ed.ac.uk/teaching/courses/asr/labs-2020.html

1.4. Evaluation Method

The number of forward computations per wav FCPW

1.4.1. ACCURACY

This report is about a hybrid architecture speech recognition system, which mainly studies the impact of WFST on the overall system. This speech recognition system inputs audio and outputs a sequence of words. Therefore, as with traditional ASR systems, the accuracy rate is expressed using the standard Word Error Rate (WER) metric. The WER shown in this report is the average WER of all samples, which is the total number of errors divided by the total number of words. In the following tables, WER also means WER under the initial structure, compared to WER (U) means WER under the structure combined with unigram, and WER (B) means WER under the structure combined with bigram.

1.4.2. MEMORY

In this task, the memory usage is mainly on WFST storage, which is closely related to the number of states and arcs. Therefore, the number of states and arcs is used as a measure of the memory space used. Since the above matrices are only related to the structure of the WFST and have nothing to do with weights and search methods, Table 1 can represent all the memory-related results comparisons for all experiments.

1.4.3. SPEED

The speed of the Viterbi decoder is crucial for an ASR system, which affects the response speed of the overall system. This report uses the number of forward computations per wav (FCPW) to represent decoding speed. FCPW counts when the system need to compute the likelihood along an arc in the WFST, which closely correlated with decoding speed. Although $cProfile$ [3] was used in the experiment to feedback the running time and number of calls of each function, this report only uses FCPW to reflect the speed for two reasons. First, the *decode*() function takes much longer to run than the *backtrace*() function. For example, when running the *baseline* experiment, it took a total of 1905.854 seconds, where the cumtime of the *decode*() function was 1684.631 s and the cumtime of the *backtrace*() function was 0.934 s. Secondly, because this experiment was run during the peak hours of remote learning in schools, the hardware, especially the CPU, allocated to many experiments was not the same, which made it difficult to reflect

[3] https://docs.python.org/3.2/library/profile.html

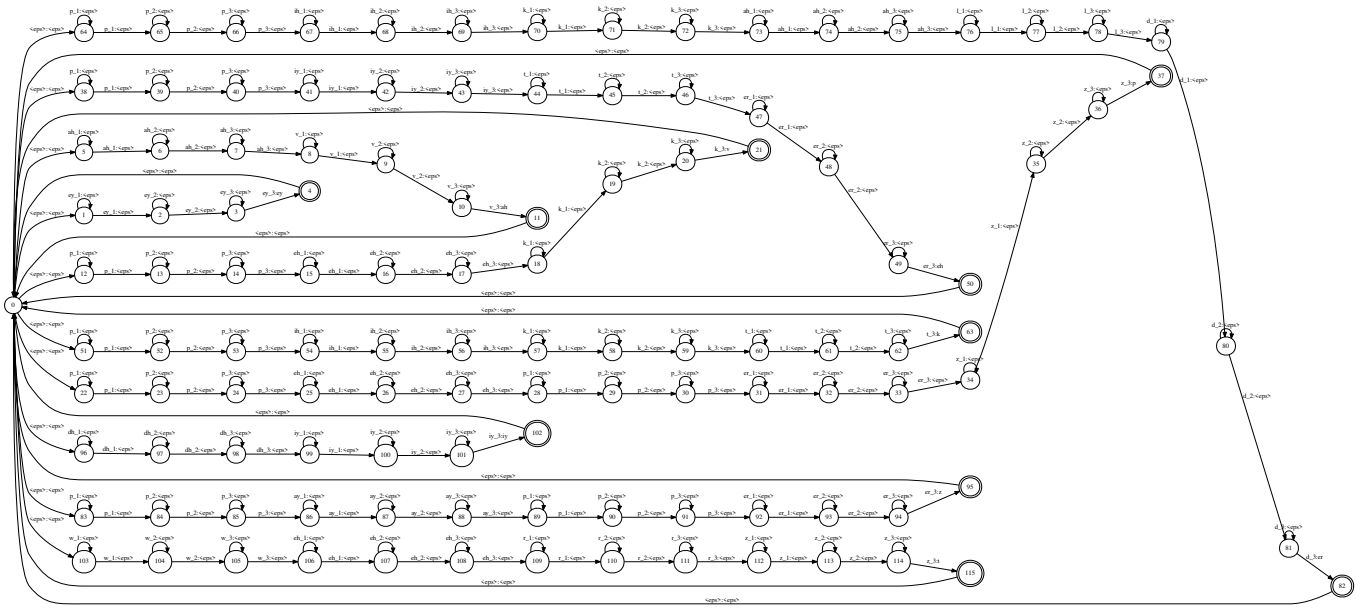

Figure 1. Figure shows the baseline structure.

EXPERIMENT	NEXT	SELF LOOP	WER
N0.9S0.1	0.9	0.1	1.0905
N0.7S0.3	0.7	0.3	0.81305
N0.5S0.5	0.5	0.5	0.75471
N0.4S0.6	0.4	0.6	0.73588
N0.3S0.7	0.3	0.7	0.71750
N0.2S0.8	0.2	0.8	0.69959
BASELINE			0.77768

Table 2. Result for next weights and self loop weights experiment based on baseline structure.

the relevant speed.

1.5. Baseline result

WER of the *baseline* experiment is shown in Table 2, 3, 4, 5, 6, 7, 8 for comparison. Memory and speed result is shown in Table 1. The results show that the system under the *baseline* setting has a relatively high error rate. However, because the *baseline* setting system is relatively simple, the memory space used is relatively small and has a relatively quick running time.

2. System tuning

This section will adjust the system parameters and improve the system structure to achieve higher recognition capabilities. Because Section 2.1, Section 2.2 and Section 2.3 only adjust the weights, the WFST structure is the same as the *baseline*, so the memory and speed issues are not discussed in these two sections.

2.1. Next weights and self loop weights

First, for each non-final state, there is a self loop probability *sl_weight* to return to its own state and a next probability *next_weight* to enter the next state. This section explores the impact of different *sl_weight* and *next_weight* on recognition, where the sum of *sl_weight* and *next_weight* is 1.

Table 2 summaries the result for this experiment based on baseline structure. The results show that experiments with *sl_weight* greater than 0.5 and *next_weight* less than 0.5 are better than baseline. Moreover, as *sl_weight* becomes larger and *next_weight* becomes smaller, the effect of system recognition is better. The best setting in this group of experiments is $n0.2s0.8$ which sets *next_weight* = 0.2 and *sl_weight* = 0.8.

2.2. Final weights and back weights

The previous section mainly considered the non-final state, and this section will consider the final state. In the *baseline* structure shown in Figure 1, the final state has a final probability $final\_weight$ of ending the entire recognition, and a probability *back_weight* to continue, that is, to return to the starting point. This section explores the impact of different $final\_weight$ and *back_weight* settings on recognition, where the sum of $final\_weight$ and *back_weight* is 1.

EXPERIMENT	FINAL	BACK	WER
F0.7B0.3	0.7	0.3	0.69224
F0.6B0.4	0.6	0.4	0.69407
F0.5B0.5	0.5	0.5	0.69545
F0.2B0.8	0.2	0.8	0.69959
N0.2S0.8(BASELINE)			0.69959
BASELINE			0.77768

Table 3. Result for final weights and back weights experiment based on baseline structure. Set next weight = 0.2 and self loop weight = 0.8.

EXPERIMENT	WER	WER(UNIGRAM)	IMPRO.
N0.7S0.3	0.81305	0.80937	0.00368
N0.5S0.5	0.75471	0.75425	0.00046
N0.4S0.6	0.73588	0.73588	0
N0.3S0.7	0.71750	0.71704	0.00046
N0.2S0.8	0.69959	0.69867	0.00092
F0.7B0.3	0.69224	0.69132	0.00092
F0.5B0.5	0.69545	0.69270	0.00275
F0.2B0.8	0.69959	0.68535	0.01424
BASELINE			0.77768

Table 4. Result for unigram experiments based on baseline structure.

Table 3 indicates the result for this experiment based on baseline structure with *next_weight* = 0.2 and *sl_weight* = 0.8 which is the best setting $n0.2s0.8$ from previous section. Thus, $n0.2s0.8$ is another baseline for this part of experiments. The results show that as the $final\_weight$ increases, the *back_weight* decreases, the recognition performance of the system slightly increases. However, the increase was minimal. It can be seen that adjusting these two parameters has no significant effect on system optimization.

2.3. Unigram

In previous experiments, the probability *word_weight* from the start state to the state where each word starts in WFST is equal. This is just a kind of naive hypothesis, which is counter intuitive. Therefore, this section will explore the incorporation of unigram into the *baseline* structure. Specifically, *word_weight* is based on the probability of each word in the unigram model which is conducted from the transcriptions in the dataset.

Table 4 shows the result for a group of experiments discovered before based on baseline structure with or without unigram. From the results, after using unigram, the results of various experiments have improved slightly. However, except for the $f0.2b0.8$ experiment, which increased by 0.01424, the improvement of other experiments were very small. It can be seen that for this dataset, unigram is not very useful. This may be related to the dataset itself. It is not difficult to see that some examples from the dataset given in this assignment has no practical meaning, and the vocabulary is limited. Thus, it cannot reflect the real speech recognition scene.

EXPERIMENT	BACK	SILENCE	WER	WER(U)
B0.8SIL0.2	0.8	0.2	0.61231	0.61001
B0.5SIL0.5	0.5	0.5	0.60955	0.60772
B0.2SIL0.8	0.2	0.8	0.60726	0.60542
B0.1SIL0.9	0.1	0.9		0.60037
B0.01SIL0.99	0.01	0.99		0.59577
N0.2S0.8(BASELINE)			0.69959	
BASELINE			0.77768	

Table 5. Result for back weights and silence weights experiment based on silence structure with or without unigram. Set next weight = 0.2 and self loop weight = 0.8.

2.4. Silence

2.4.1. STRUCTURE

This section will explore the influence of adding silence states between words in the WFST, which can handle the case where there are pauses between words in the speech. This refers to *silence*(*unigram*) structure in Table 1 and Figure 2 shows the detailed silence structure with unigram. The difference from the *baseline* structure is that in *silence*(*unigram*) the state at the end of each word no longer has the $final\_weight$ probability of final recognition, but retains the *back_weight* probability of returning the starting state, adding the probability of entering silence states, *sil_weight*. The silence states consist of five states from left to right. An end state has been added to silence states, which contains a probability *sli_back_weight* to return to the starting state and a probability $final\_weight$ to end the entire recognition process.

The sum of *back_weight* and *sil_weight* and the sum of $final\_weight$ and *sli_back_weight* both are 1. Based on such a structure, the following experiments will adjust these two pairs of parameters separately to achieve a relatively good silence adding structure model.

2.4.2. BACK WEIGHTS AND SILENCE WEIGHTS

Table 5 indicates the result for back weights and silence weights experiment based on silence structure with or without unigram setting *next_weight* = 0.2 and *sl_weight* = 0.8 which is the best setting $n0.2s0.8$ from Section 2.1.

First of all, the results show that after adding silence states, the system has been greatly improved for recognition performance. Compared with the *baseline* of 0.77768 and $n0.2s0.8$ of 0.69959, the *silence* structure system can reach a WER of about 0.60, without the addition of unigram.

Second, as the *back_weight* decreases and *sil_weight* increases, the better the system performance, even reaching the maximum when *sil_weight* = 0.99. This shows that basically all samples have better matching ability on the *silence* structure. This is in line with our original intention of designing this structure, which is to handle the case of

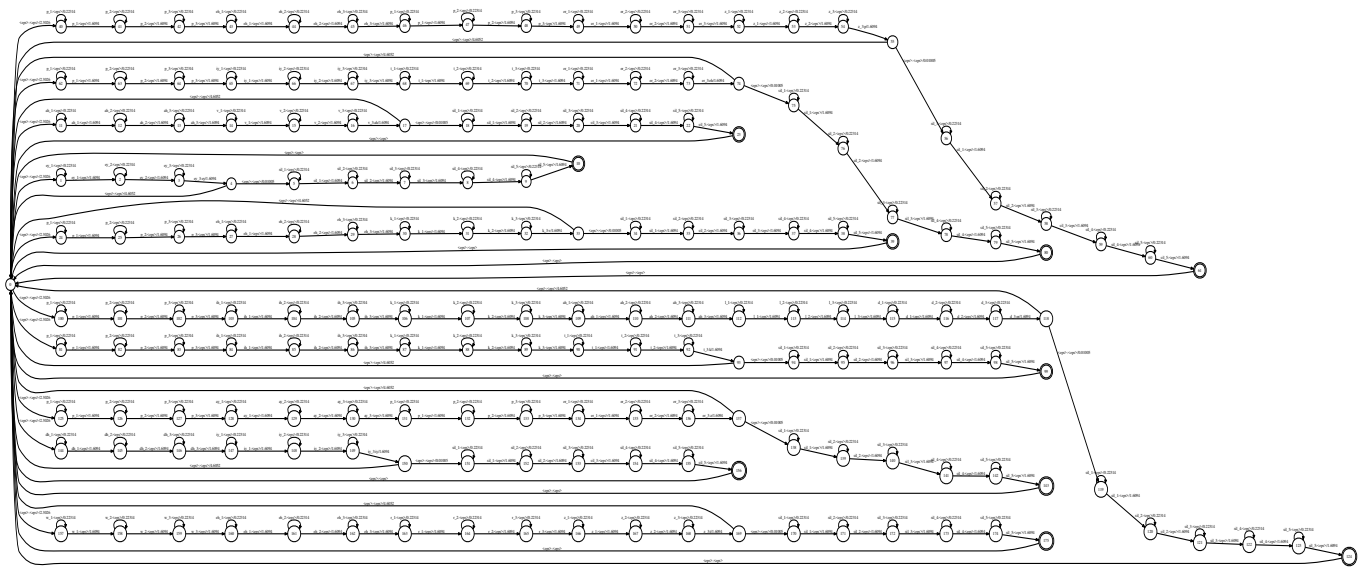

Figure 2. Figure shows the silence structure (unigram).

EXPERIMENT	FINAL	SILENCE BACK	WER(U)
F0.8SILB0.2	0.8	0.2	0.59348
F0.5SILB0.5	0.5	0.5	0.59577
F0.2SILB0.8	0.2	0.8	0.59348
B0.01SIL0.99(BASELINE)			0.59577
BASELINE			0.77768

Table 6. Result for final weights and silence back weights experiment based on silence structure with unigram. Set next weight = 0.2, self loop weight = 0.8, back weight = 0.01 and silence weight = 0.99.

pauses in sentences. This has universal value for all audio.

Third, adding unigram to the structure can also improve the performance of the *silence* structure. In particular, for $b0.01sil0.99$ experiment, its WER drops to 0.59577.

2.4.3. FINAL WEIGHTS AND SILENCE BACK WEIGHTS

The previous section adjusted the back weights and silence weights so that it reached 0.59577 in experiment $b0.01sil0.99$. In this section, based on experiment $b0.01sil0.99$, the final weights and silence back weights will be adjusted. Since the role of unigram on the *silence* structure has been verified, this section uses unigram directly.

Table 6 summaries the result for this experiment based on silence structure with unigram setting *next_weight* = 0.2, *sl_weight* = 0.8, *back_weight* = 0.01, *sil_weight* = 0.99, which is the best setting $b0.01sil0.99$ from previous section. The results show that, as in Section 2.2, the change in final weights and silence back weights has little effect on the overall recognition effect of the system. However, the experiment $f0.2silb0.8still$ reached this best WER, 0.59348, although this is a small difference from $b0.01sil0.99$.

2.4.4. MEMORY AND SPEED

The previous experiments on weight adjustment only discussed WER, and here this section will discuss memory and speed. Since memory and speed are only related to structures according to the previous mentioned evaluation method, the *baseline* and *silence*(*unigram*) structures will be compared here.

According to Table 1, the *silence*(*unigram*) structure is basically 1.5 times the *baseline* in memory usage. In terms of time spent, the time required for *silence*(*unigram*) is basically 1.5 times that of the *baseline*. However, in terms of performance, it can be reduced from 0.77768 to 0.59348. In general, the cost performance of the *silence*(*unigram*) structure is still higher than the *baseline*. Follow-up experiments will study pruning, which has the possibility of spending less time on the *silence*(*unigram*) structure but achieving much higher performance than the *baseline*.

Experiment	threshold	WER(U)	FCPW
Drop1e3	1e3	0.76390	73761.4
Drop3e3	3e3	0.65181	116178.2
Drop5e3	5e3	0.62563	132481.2
Drop8e3	8e3	0.60083	144947.2
Drop1e4	1e4	0.59302	152647.8
Drop1e6	1e6	0.59577	160839.8
Drop1e8	1e8	0.59577	160839.8
b0.01sil0.09(baseline)		0.59577	160839.8
baseline		0.77768	109421.5

Table 7. Result for Pruning - Drop small probabilities experiment based on silence structure with unigram. Set next weight = 0.2, self loop weight = 0.8, back weight = 0.01 and silence weight = 0.99.

Experiment	Beam Width	WER(U)	FCPW
Beam10	10	0.80937	9078.3
Beam20	20	0.68213	19227.5
Beam30	30	0.65044	29507.3
Beam40	40	0.63987	39661.1
Beam50	50	0.61966	49507.7
Beam60	60	0.61093	59401.5
Beam70	70	0.60542	69159.4
Beam80	80	0.60450	78628.7
Beam90	90	0.60175	88210.4
Beam100	100	0.59853	97774.8
b0.01sil0.09(baseline)		0.59577	160839.8
baseline		0.77768	109421.5

Table 8. Result for Pruning - Beam search experiment based on silence structure with unigram. Set next weight = 0.2, self loop weight = 0.8, back weight = 0.01 and silence weight = 0.99.

Beam Width	WER(U)	FCPW(U)	WER(B)	FCPW(B)
50	0.61966	49507.7	0.63390	40131.0
70	0.60542	69159.4	0.61047	58598.1
100	0.59853	97774.8	0.59945	85863.2
ALL	0.59577	160839.8	0.59394	160839.8

Table 9. Result for Bigram experiment based on silence structure comparing to unigram. Set next weight = 0.2, self loop weight = 0.8, back weight = 0.01 and silence weight = 0.99.

3. Pruning

This section will investigate two pruning methods for Viterbi algorithm avoiding propagating computations on unlikely paths including dropping small probabilities and beam search.

3.1. Drop small probabilities

Dropping small probabilities is to add a judgment of the previous step's likelihood during using Viterbi algorithm. If the likelihood is greater than a fixed value *pruning_threshold*, this path is not considered (because it is Negative Log-Likelihood (NLL), so it is greater than, and the value of *pruning_threshold* is also directly expressed by NLL).

Table 7 summaries the result for this experiment based on silence structure with unigram setting *next_weight* = 0.2, *sl_weight* = 0.8, *back_weight* = 0.01, *sil_weight* = 0.99, which is the best setting *b*0.01*sil*0.99 from Section 2.4.2. From the results, as the threshold is gradually reduced, the time required for operation is gradually reduced, and the system performance is gradually weakened. It can be found form the table that it is more suitable when the *pruning_threshold* = 8*e*3 because under this condition, the system performance is not reduced much, but the operation speed is improved.

Although this method achieves a certain degree of improvement in speed without significant loss of performance. However, due to method defects, it can not improve a lot of speed, and tuning the threshold is extremely difficult. This is mainly because the probability of different paths and different depths may be different in magnitude. Using the same threshold may cause more wasted computations on some paths.

3.2. Beam search

Similar to dropping small probabilities, beam search also discards for lower probability in Viterbi algorithm. But unlike dropping small probabilities setting a fixed probability *pruning_threshold*, beam search maintains a beam whose width is *beam_width*. For all the probabilities of the previous step, first sort them, put the top *beam_width* paths into beam, and then look for the probability of the previous step in beam. This method of taking the window as the core idea makes it extremely efficient to reduce the calculation without extremely loss.

Table 8 shows the result for this experiment based on silence structure with unigram setting *next_weight* = 0.2, *sl_weight* = 0.8, *back_weight* = 0.01, *sil_weight* = 0.99, which is the best setting *b*0.01*sil*0.99 from Section 2.4.2. From the results, *beam_width* has a very stable control over the speed of operations. What's exciting is that when *beam_width* = 100, the system's recognition ability has almost no change, but the speed is almost doubled, which is faster than the *baseline* experiment. Of course, *beam_width* = 70 is also an ideal threshold. At that point, the performance loss is slight, but the speed is increased by 2 to 3 times.

Compared with the strategy of the previous section, the effect of beam search is far better than the previous results, which again verifies the conclusion of the previous section and the advantages of beam search.

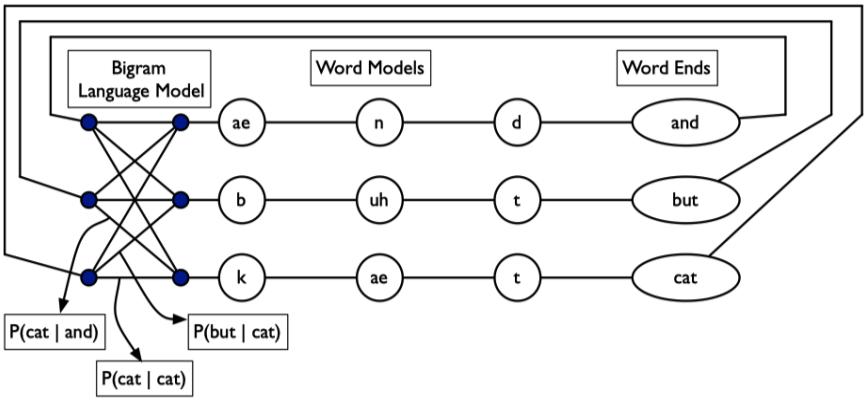

Figure 3. Connected word recognition with a bigram language model. (Renals & Hain, 2010)

4. Advanced topics

4.1. Bigram

According to (Renals & Hain, 2010), bigram language model is easily incorporate with HMM but trigrams require a word history to be maintained. For this reason, this part only implements bigram language model with the *silence* structure. Figure 3 shows how bigram language model is incorporate with word recognition system and *silence*(*bigram*) in Table 1 shows the states number and arcs number.

Table 9 shows the result for this experiment based on silence structure comparing to unigram with different beam widths setting $next\_weight = 0.2$, $sl\_weight = 0.8$, $back\_weight = 0.01$, $sil\_weight = 0.99$, which is the best setting $b0.01sil0.99$ from Section 2.4.2. The results show that bigram is better than unigram, it improved from 0.59577 to 0.59394, but the improvement is very slight. Under different *beam_width* settings, bigram can also speed up processing without greatly affecting performance, which once again validates the advantages of beam search.

In terms of computing speed, bigram and unigram are similar. But in storage space, bigram needs more space to store states and arcs. Based on the current results, there is no advantage to using bigram in this dataset. This is actually related to the dataset itself. First, many samples don't have a very general grammer, they are more like randomly generated. Second, the speech and transcription of some samples do not correspond, which will be mentioned in the next section.

References

Renals, Steve and Hain, Thomas. *Speech Recognition*, chapter 12, pp. 297–332. John Wiley Sons, Ltd, 2010. ISBN 9781444324044. doi: 10.1002/9781444324044.ch12. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781444324044.ch12.