Experiment 2: CS-539

Sanad Saha 933620612

1 Training Language Models

1. Train unigram, bigram, and trigram character language models. These models must assign non-zero probability to any sequence. Store these models in Carmel's WFSA format. Evaluate your models on text.txt. Try improve your models. Your grade will depend in large part on the entropies of your models on the blind test data.
Name your models: unigram.wfsa, bigram.wfsa, trigram.wfsa.
Answer:

I've trained the unigram, bigram and trigram character language models on the train.txt using make_unigram.py, make_bigram.py and make_trigram.py respectively. First I trained unigram, bigram and trigram model without smoothing and tested them on test.txt. The language models without smoothing wasn't able to generate derivations for all input lines in the test.txt file. I used the following command for training the models:

```
python3 make_<language_model>.py > <language_model>.wfsa
```

Then I added smoothing; First I did the laplace smoothing and then also tried the Lidstone smoothing (add-k smoothing). I'm providing the language models with smoothing with the report.

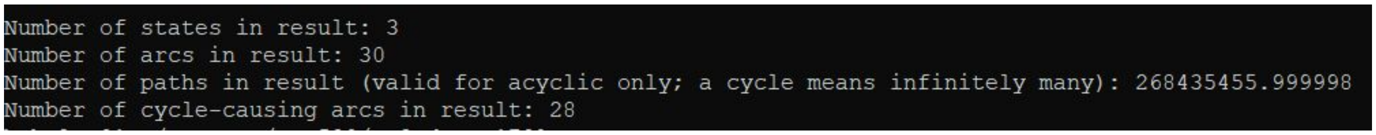
2. What are the corpus probabilities your three models assign to the test data? What are the respective (cross) entropies?
Answer:

Running our language models over the test data we got the following probabilities and entropies.

Language Models	Corpus Probability	Entropy per input symbol	Entropy Per-line
Unigram	2^-39012	4.11519	390.12
Bigram	2^-32448.4	3.42283	324.484
Trigram	2^-27640.2	2.91564	276.402

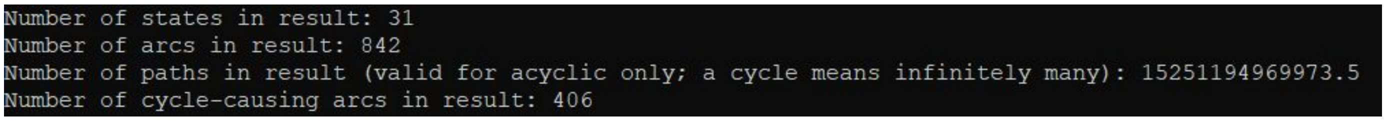
3. What are the final sizes of your WFSAs in states and transitions.
<u>Answer:</u>

Unigram: [3 states and 30 transitions]



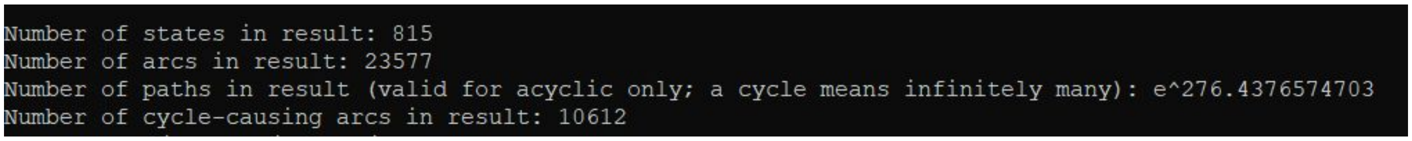
```
Number of states in result: 3
Number of arcs in result: 30
Number of paths in result (valid for acyclic only; a cycle means infinitely many): 268435455.999998
Number of cycle-causing arcs in result: 28
```

Bigram: [31 states and 843 transitions]



```
Number of states in result: 31
Number of arcs in result: 842
Number of paths in result (valid for acyclic only; a cycle means infinitely many): 15251194969973.5
Number of cycle-causing arcs in result: 406
```

Trigram: [815 states and 23577 transitions]

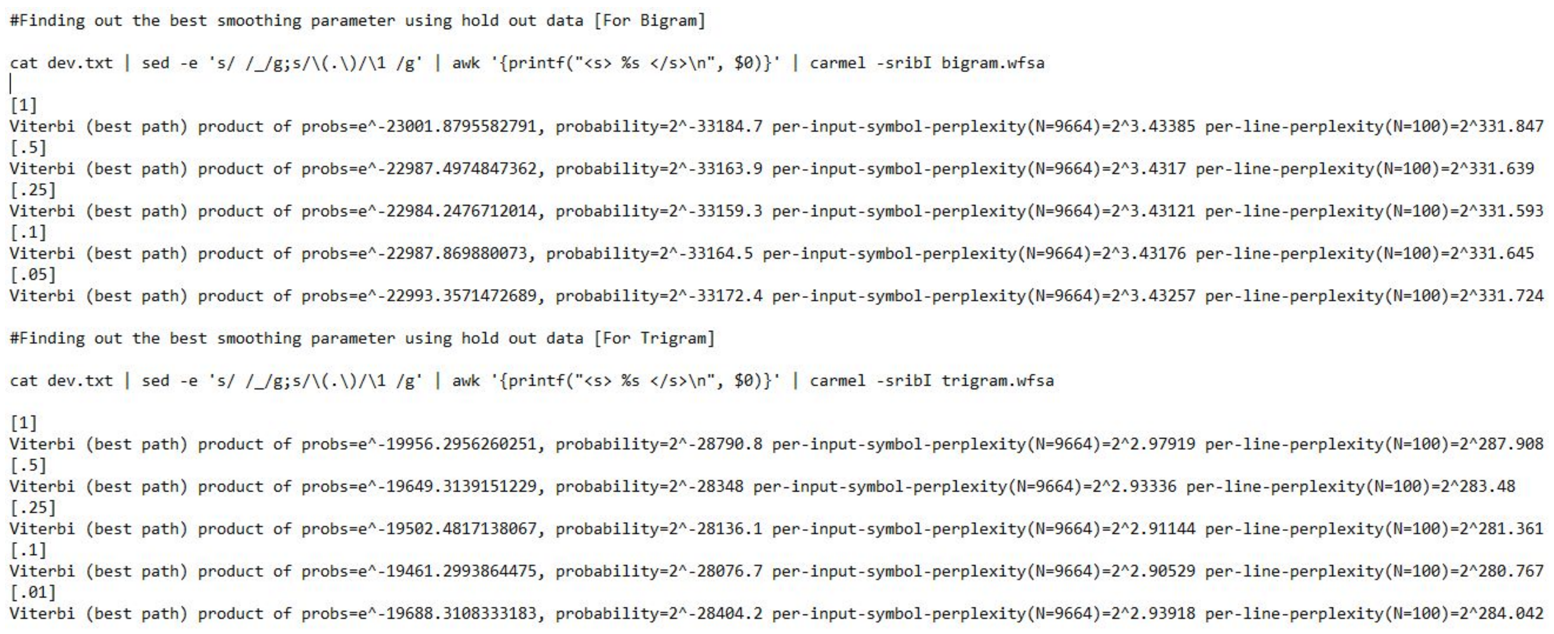


```
Number of states in result: 815
Number of arcs in result: 23577
Number of paths in result (valid for acyclic only; a cycle means infinitely many): e^276.4376574703
Number of cycle-causing arcs in result: 10612
```

4. Include a description of your smoothing method. Your description should include the algorithms, how you used the held-out data, and what (if any) experiments you did before settling on your solution.

Answer:

For smoothing we used both Laplace smoothing and Lidstone smoothing. In laplace smoothing 1 is added to occurrences which didn't appear on the test data. And in the Lidstone smoothing values less than 1 are added to the cases which were unseen in the test data. We tested our bigram model with lidstone smoothing on the held-out data (dev.txt) for values 0.5, 0.25, 0.1,

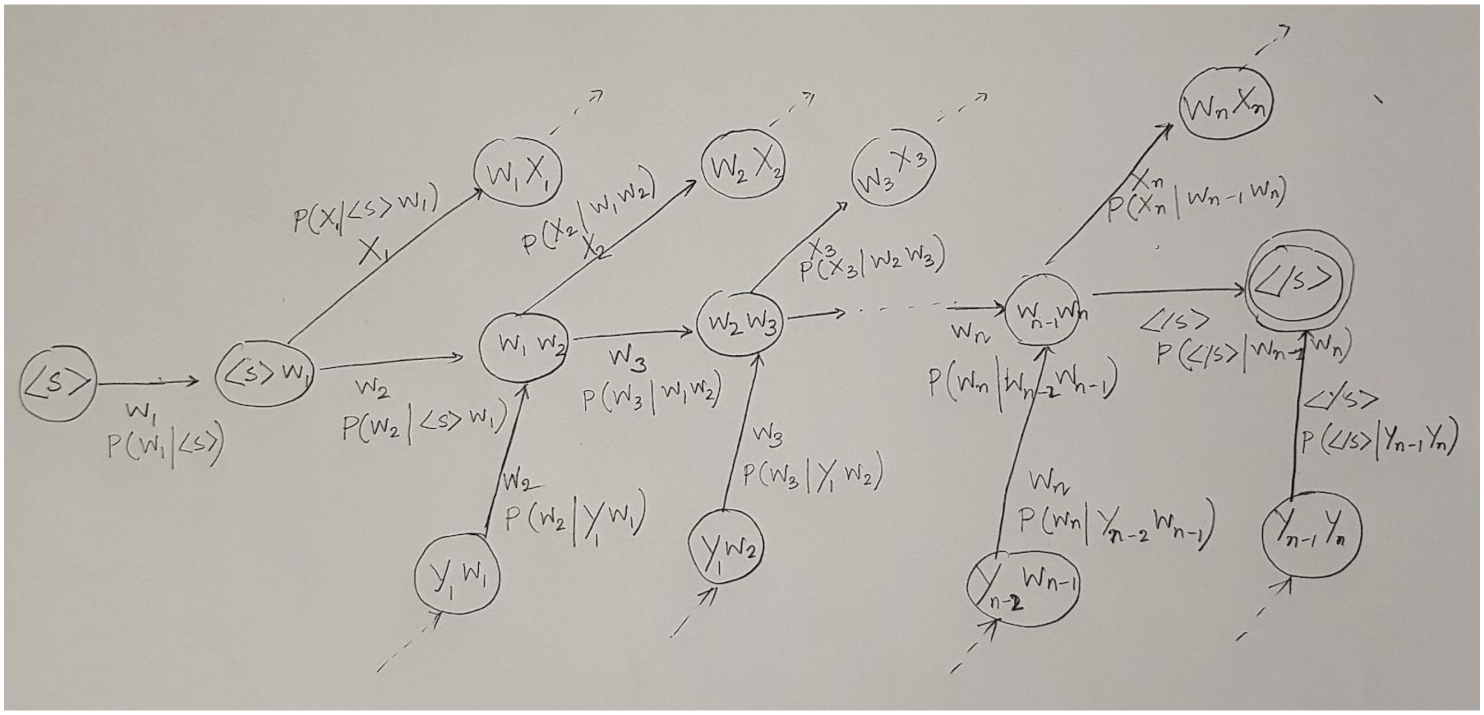


```
#Finding out the best smoothing parameter using hold out data [For Bigram]

cat dev.txt | sed -e 's/ /_/g;s/\(.\)/\1 /g' | awk '{printf("<s> %s </s>\n", $0)}' | carmel -sribI bigram.wfsa

[1]
Viterbi (best path) product of probs=e^-23001.8795582791, probability=2^-33184.7 per-input-symbol-perplexity(N=9664)=2^3.43385 per-line-perplexity(N=100)=2^331.847
[.5]
Viterbi (best path) product of probs=e^-22987.4974847362, probability=2^-33163.9 per-input-symbol-perplexity(N=9664)=2^3.4317 per-line-perplexity(N=100)=2^331.639
[.25]
Viterbi (best path) product of probs=e^-22984.2476712014, probability=2^-33159.3 per-input-symbol-perplexity(N=9664)=2^3.43121 per-line-perplexity(N=100)=2^331.593
[.1]
Viterbi (best path) product of probs=e^-22987.869880073, probability=2^-33164.5 per-input-symbol-perplexity(N=9664)=2^3.43176 per-line-perplexity(N=100)=2^331.645
[.05]
Viterbi (best path) product of probs=e^-22993.3571472689, probability=2^-33172.4 per-input-symbol-perplexity(N=9664)=2^3.43257 per-line-perplexity(N=100)=2^331.724

#Finding out the best smoothing parameter using hold out data [For Trigram]

cat dev.txt | sed -e 's/ /_/g;s/\(.\)/\1 /g' | awk '{printf("<s> %s </s>\n", $0)}' | carmel -sribI trigram.wfsa

[1]
Viterbi (best path) product of probs=e^-19956.2956260251, probability=2^-28790.8 per-input-symbol-perplexity(N=9664)=2^2.97919 per-line-perplexity(N=100)=2^287.908
[.5]
Viterbi (best path) product of probs=e^-19649.3139151229, probability=2^-28348 per-input-symbol-perplexity(N=9664)=2^2.93336 per-line-perplexity(N=100)=2^283.48
[.25]
Viterbi (best path) product of probs=e^-19502.4817138067, probability=2^-28136.1 per-input-symbol-perplexity(N=9664)=2^2.91144 per-line-perplexity(N=100)=2^281.361
[.1]
Viterbi (best path) product of probs=e^-19461.2993864475, probability=2^-28076.7 per-input-symbol-perplexity(N=9664)=2^2.90529 per-line-perplexity(N=100)=2^280.767
[.01]
Viterbi (best path) product of probs=e^-19688.3108333183, probability=2^-28404.2 per-input-symbol-perplexity(N=9664)=2^2.93918 per-line-perplexity(N=100)=2^284.042
```

0.05 and the trigram model with values 0.5, .25, 0.1, .01. And we settled for lidstone constant 0.25 for the bigram and 0.1 for the trigram. The following screenshot shows all the experiments we ran and the results we got on the held-out data.

5. Include a sketch drawing of your 3-gram model in sufficient detail that someone could replicate it. Please consider this carefully { examine your drawing after you have drawn it and evaluate whether someone (not you) could build the same WFSA you have built. We will consider it in the same light.

Answer:

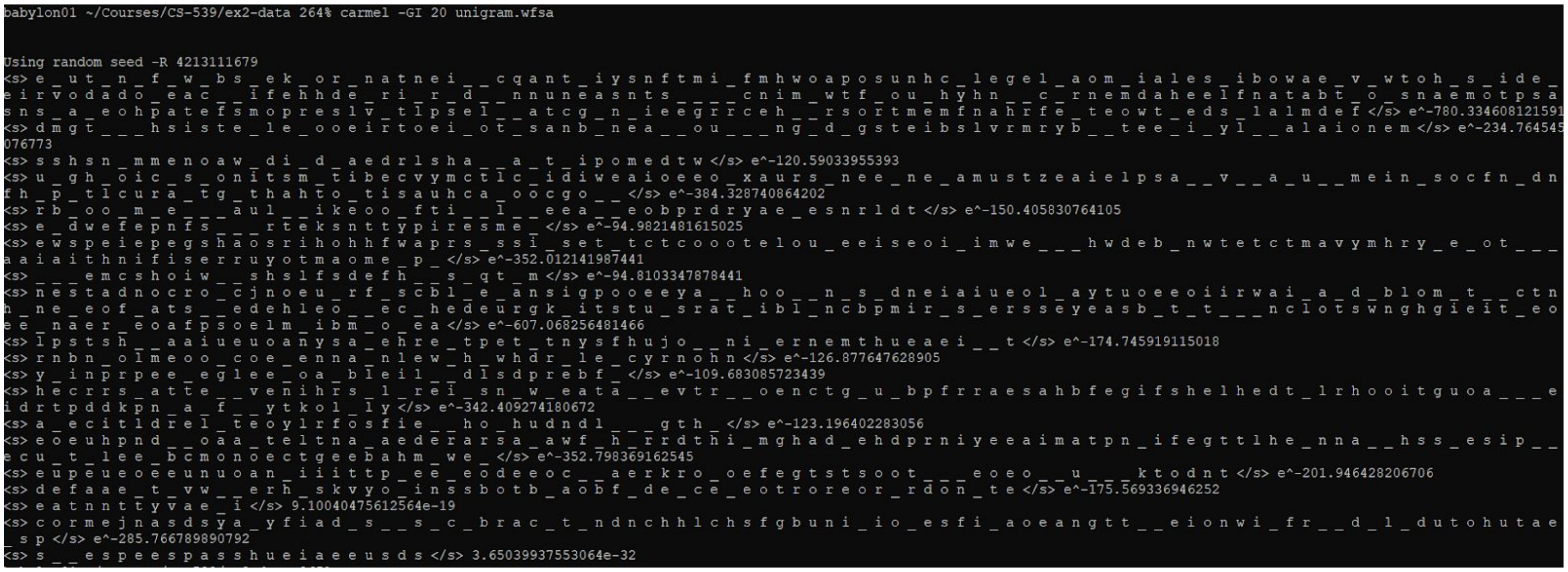


In this trigram model, <s> is the starting state. W_n (n = 1, 2, 3...) are characters a to z, - (dash) and _ (spaces). Now transition to a new state depends on past two characters. For example, from state (W_1 W_2) there are P(W_3|W_1 W_2) probability of going to state (W_2 W_3). From (W_1 W_2) it's also possible to go to other states which are represented with (W_1 X_1) where X1 is all characters aside from W2. To generalize, X_n contains all possible character aside from W_n. There are also many possible ways to reach a state (W_{n-1} W_n); which are portrayed using states (Y_{n-2} W_{n-1}). So, from state (Y_{n-2} W_{n-1}) using transition W_n it's possible to reach state (W_{n-1} W_n) with probability P(W_n|Y_{n-2} W_{n-1}). The final state is </s>. To summarize, transitions in trigram model depends on the earlier 2 letters seen. There are no outgoing states from the final state and there are no incoming states to the starting state.

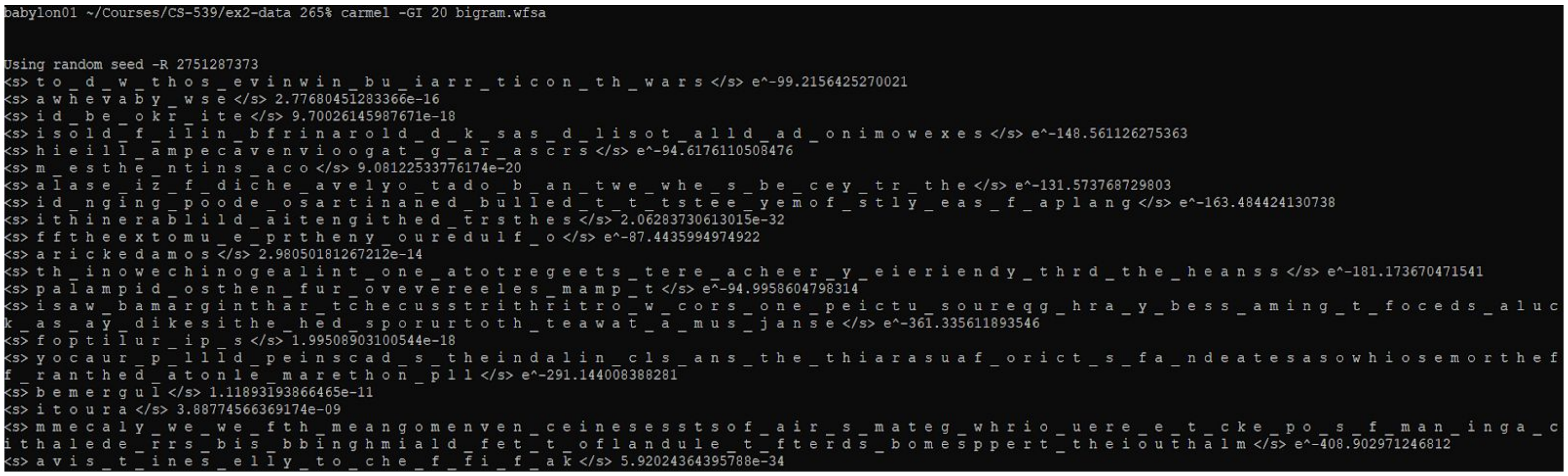
2 Using Language Models

1. Random generation from n-gram models.
Use carmel -GI 20 <your_wfsa> to stochastically generate character sequences. Show the results.
Do these results make sense? (For example, you can do the same on the above sample WFSA, and you will see the proportion of a:b is indeed about 8:1.)
Answer:

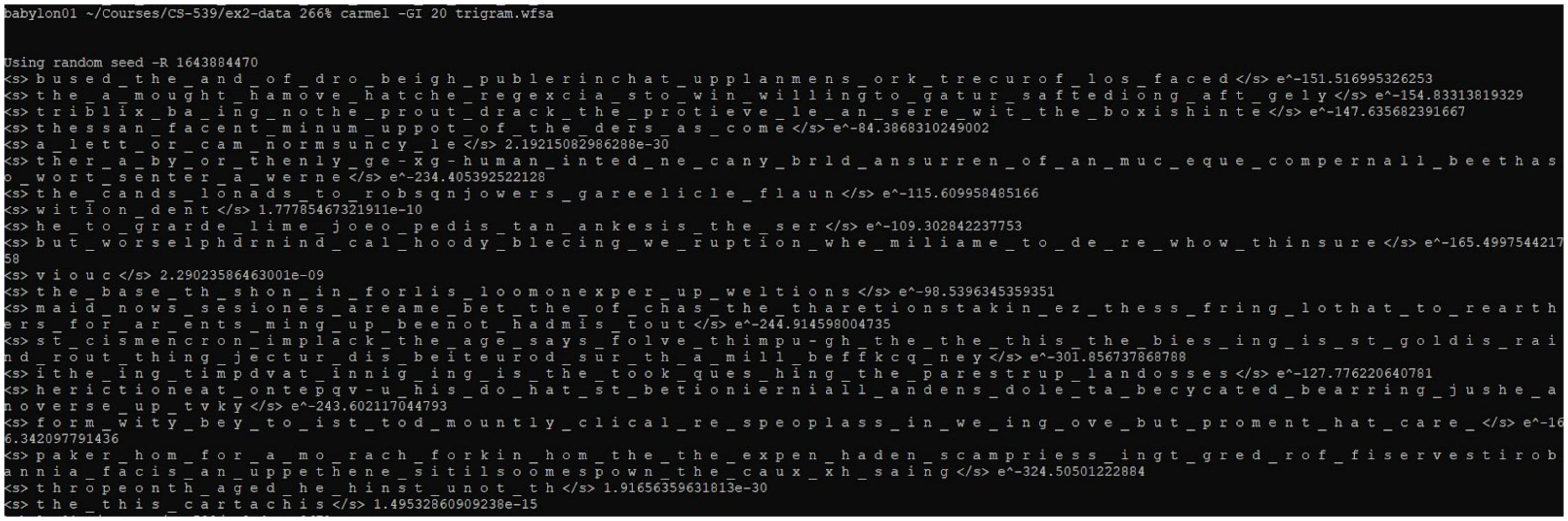
The following screenshots show the generated character sequence for our language models. carmel -GI 20 <language model.wfsa > gives us the 20 statistically generated paths. All the sequences start with <s> and ends with </s> because these two are the starting and ending state of all the language models. For the unigram model we can see that sequences starting with e, s, n because of the high frequency of these letters in our training data (also in English literature). As



```
babylon01 ~/Courses/CS-539/ex2-data 264% carmel -GI 20 unigram.wfsa


Using random seed -R 4213111679
<s> e _ u t _ n _ f _ w _ b s _ e k _ o r _ n a t n e i _ _ c q a n t _ i y s n f t m i _ f m h w o a p o s u n h c _ l e g e l _ a o m _ i a l e s _ i b o w a e _ v _ w t o h _ s _ i d e _
e i r v o d a d o _ e a c _ _ i f e h h d e _ r i _ r _ d _ _ n n u n e a s n t s _ _ _ _ c n i m _ w t f _ o u _ h y h n _ _ c _ r n e m d a h e e l f n a t a b t _ o _ s n a e m o t p s a
s n s _ a _ e o h p a t e f s m o p r e s l v _ t l p s e l _ _ a t c g _ n _ i e e g r r c e h _ _ r s o r t m e m f n a h r f e _ t e o w t _ e d s _ l a l m d e f </s> e^-780.334608121591
<s> d m g t _ _ _ h s i s t e _ l e _ o o e i r t o e i _ o t _ s a n b _ n e a _ _ o u _ _ _ n g _ d _ g s t e i b s l v r m r y b _ _ t e e _ i _ y l _ _ a l a i o n e m </s> e^-234.764545
076773
<s> s s h s n _ m m e n o a w _ d i _ d _ a e d r l s h a _ _ a _ t _ i p o m e d t w </s> e^-120.59033955393
<s> u _ g h _ o i c _ s _ o n i t s m _ t i b e c v y m c t l c _ i d i w e a i o e e o _ x a u r s _ n e e _ n e _ a m u s t z e a i e l p s a _ _ v _ _ a _ u _ _ m e i n _ s o c f n _ d n
f h _ p _ t l c u r a _ t g _ t h a h t o _ t i s a u h c a _ o o c g o _ _ </s> e^-384.328740864202
<s> r b _ o o _ m _ e _ _ _ a u l _ _ i k e o o _ f t i _ _ l _ _ e e a _ _ e o b p r d r y a e _ e s n r l d t </s> e^-150.405830764105
<s> e _ d w e f e p n f s _ _ _ r t e k s n t t y p i r e s m e _ </s> e^-94.9821481615025
<s> e w s p e i e p e g s h a o s r i h o h h f w a p r s _ s s i _ s e t _ t c t c o o o t e l o u _ e e i s e o i _ i m w e _ _ _ h w d e b _ n w t e t c t m a v y m h r y _ e _ o t _ _ _
a a i a i t h n i f i s e r r u y o t m a o m e _ p _ </s> e^-352.012141987441
<s> _ _ _ e m c s h o i w _ _ s h s l f s d e f h _ _ s _ q t _ m </s> e^-94.8103347878441
<s> n e s t a d n o c r o _ c j n o e u _ r f _ s c b l _ e _ a n s i g p o o e e y a _ _ h o o _ _ n _ s _ d n e i a i u e o l _ a y t u o e e o i i r w a i _ a _ d _ b l o m _ t _ _ c t n
h _ n e _ e o f _ a t s _ _ e d e h l e o _ _ e c _ h e d e u r g k _ i t s t u _ s r a t _ i b l _ n c b p m i r _ s _ e r s s e y e a s b _ t _ t _ _ _ n c l o t s w n g h g i e i t _ e o
e e _ n a e r _ e o a f p s o e l m _ i b m _ o _ e a </s> e^-607.068256481466
<s> l p s t s h _ _ a a i u e u o a n y s a _ e h r e _ t p e t _ t n y s f h u j o _ _ n i _ e r n e m t h u e a e i _ _ t </s> e^-174.745919115018
<s> r n b n _ o l m e o o _ c o e _ e n n a _ n l e w _ h _ w h d r _ l e _ c y r n o h n </s> e^-126.877647628905
<s> y _ i n p r p e e _ e g l e e _ o a _ b l e i l _ _ d l s d p r e b f _ </s> e^-109.683085723439
<s> h e c r r s _ a t t e _ _ v e n i h r s _ l _ r e i _ s n _ w _ e a t a _ _ e v t r _ _ o e n c t g _ u _ b p f r r a e s a h b f e g i f s h e l h e d t _ l r h o o i t g u o a _ _ _ e
i d r t p d d k p n _ a _ f _ _ y t k o l _ l y </s> e^-342.409274180672
<s> a _ e c i t l d r e l _ t e o y l r f o s f i e _ _ h o _ h u d n d l _ _ _ g t h _ </s> e^-123.196402283056
<s> e o e u h p n d _ _ o a a _ t e l t n a _ a e d e r a r s a _ a w f _ h _ r r d t h i _ m g h a d _ e h d p r n i y e e a i m a t p n _ i f e g t t l h e _ n n a _ _ h s s _ e s i p _ _
e c u _ t _ l e e _ b c m o n o e c t g e e b a h m _ w e _ </s> e^-352.798369162545
<s> e u p e u e o e e u n u o a n _ i i i t t p _ e e _ e o d e e o c _ _ a e r k r o _ o e f e g t s t s o o t _ _ _ e o e o _ _ u _ _ _ k t o d n t </s> e^-201.946428206706
<s> d e f a a e _ t _ v w _ _ e r h _ s k v y o _ i n s s b o t b _ a o b f _ d e _ c e _ e o t r o r e o r _ r d o n _ t e </s> e^-175.569336946252
<s> e a t n n t t y v a e _ i </s> 9.10040475612564e-19
<s> c o r m e j n a s d s y a _ y f i a d _ s _ _ s _ c _ b r a c _ t _ n d n c h h l c h s f g b u n i _ i o _ e s f i _ a o e a n g t t _ _ e i o n w i _ f r _ _ d _ l _ d u t o h u t a e
_ s p </s> e^-285.766789890792
<s> s _ _ e s p e e s p a s s h u e i a e e u s d s </s> 3.65039937553064e-32
```



```
babylon01 ~/Courses/CS-539/ex2-data 265% carmel -GI 20 bigram.wfsa


Using random seed -R 2751287373
<s> t o _ d _ w _ t h o s _ e v i n w i n _ b u _ i a r r _ t i c o n _ t h _ w a r s </s> e^-99.2156425270021
<s> a w h e v a b y _ w s e </s> 2.77680451283366e-16
<s> i d _ b e _ o k r _ i t e </s> 9.70026145987671e-18
<s> i s o l d _ f _ i l i n _ b f r i n a r o l d _ d _ k _ s a s _ d _ l i s o t _ a l l d _ a d _ o n i m o w e x e s </s> e^-148.561126275363
<s> h i e i l l _ a m p e c a v e n v i o o g a t _ g _ a r _ a s c r s </s> e^-94.6176110508476
<s> m _ e s t h e _ n t i n s _ a c o </s> 9.08122533776174e-20
<s> a l a s e _ i z _ f _ d i c h e _ a v e l y o _ t a d o _ b _ a n _ t w e _ w h e _ s _ b e _ c e y _ t r _ t h e </s> e^-131.573768729803
<s> i d _ n g i n g _ p o o d e _ o s a r t i n a n e d _ b u l l e d _ t _ t _ t s t e e _ y e m o f _ s t l y _ e a s _ f _ a p l a n g </s> e^-163.484424130738
<s> i t h i n e r a b l i l d _ a i t e n g i t h e d _ t r s t h e s </s> 2.06283730613015e-32
<s> f f t h e e x t o m u _ e _ p r t h e n y _ o u r e d u l f _ o </s> e^-87.4435994974922
<s> a r i c k e d a m o s </s> 2.98050181267212e-14
<s> t h _ i n o w e c h i n o g e a l i n t _ o n e _ a t o t r e g e e t s _ t e r e _ a c h e e r _ y _ e i e r i e n d y _ t h r d _ t h e _ h e a n s s </s> e^-181.173670471541
<s> p a l a m p i d _ o s t h e n _ f u r _ o v e v e r e e l e s _ m a m p _ t </s> e^-94.9958604798314
<s> i s a w _ b a m a r g i n t h a r _ t c h e c u s s t r i t h r i t r o _ w _ c o r s _ o n e _ p e i c t u _ s o u r e q g _ h r a _ y _ b e s s _ a m i n g _ t _ f o c e d s _ a l u c
k _ a s _ a y _ d i k e s i t h e _ h e d _ s p o r u r t o t h _ t e a w a t _ a _ m u s _ j a n s e </s> e^-361.335611893546
<s> f o p t i l u r _ i p _ s </s> 1.99508903100544e-18
<s> y o c a u r _ p _ l l l d _ p e i n s c a d _ s _ t h e i n d a l i n _ c l s _ a n s _ t h e _ t h i a r a s u a f _ o r i c t _ s _ f a _ n d e a t e s a s o w h i o s e m o r t h e f
f _ r a n t h e d _ a t o n l e _ m a r e t h o n _ p l l </s> e^-291.144008388281
<s> b e m e r g u l </s> 1.11893193866465e-11
<s> i t o u r a </s> 3.88774566369174e-09
<s> m m e c a l y _ w e _ w e _ f t h _ m e a n g o m e n v e n _ c e i n e s e s s t s o f _ a i r _ s _ m a t e g _ w h r i o _ u e r e _ e _ t _ c k e _ p o _ s _ f _ m a n _ i n g a _ c
i t h a l e d e _ r r s _ b i s _ b b i n g h m i a l d _ f e t _ t _ o f l a n d u l e _ t _ f t e r d s _ b o m e s p p e r t _ t h e i o u t h a l m </s> e^-408.902971246812
<s> a v i s _ t _ i n e s _ e l l y _ t o _ c h e _ f _ f i _ f _ a k </s> 5.92024364395788e-34
```



```
babylon01 ~/Courses/CS-539/ex2-data 266% carmel -GI 20 trigram.wfsa


Using random seed -R 1643884470
<s> b u s e d _ t h e _ a n d _ o f _ d r o _ b e i g h _ p u b l e r i n c h a t _ u p p l a n m e n s _ o r k _ t r e c u r o f _ l o s _ f a c e d </s> e^-151.516995326253
<s> t h e _ a _ m o u g h t _ h a m o v e _ h a t c h e _ r e g e x c i a _ s t o _ w i n _ w i l l i n g t o _ g a t u r _ s a f t e d i o n g _ a f t _ g e l y </s> e^-154.83313819329
<s> t r i b l i x _ b a _ i n g _ n o t h e _ p r o u t _ d r a c k _ t h e _ p r o t i e v e _ l e _ a n _ s e r e _ w i t _ t h e _ b o x i s h i n t e </s> e^-147.635682391667
<s> t h e s s a n _ f a c e n t _ m i n u m _ u p p o t _ o f _ t h e _ d e r s _ a s _ c o m e </s> e^-84.3868310249002
<s> a _ l e t t _ o r _ c a m _ n o r m s u n c y _ l e </s> 2.19215082986288e-30
<s> t h e r _ a _ b y _ o r _ t h e n l y _ g e - x g - h u m a n _ i n t e d _ n e _ c a n y _ b r l d _ a n s u r r e n _ o f _ a n _ m u c _ e q u e _ c o m p e r n a l l _ b e e t h a s
o _ w o r t _ s e n t e r _ a _ w e r n e </s> e^-234.405392522128
<s> t h e _ c a n d s _ l o n a d s _ t o _ r o b s q n j o w e r s _ g a r e e l i c l e _ f l a u n </s> e^-115.609958485166
<s> w i t i o n _ d e n t </s> 1.77785467321911e-10
<s> h e _ t o _ g r a r d e _ l i m e _ j o e o _ p e d i s _ t a n _ a n k e s i s _ t h e _ s e r </s> e^-109.302842237753
<s> b u t _ w o r s e l p h d r n i n d _ c a l _ h o o d y _ b l e c i n g _ w e _ r u p t i o n _ w h e _ m i l i a m e _ t o _ d e _ r e _ w h o w _ t h i n s u r e </s> e^-165.4997544217
58
<s> v i o u c </s> 2.29023586463001e-09
<s> t h e _ b a s e _ t h _ s h o n _ i n _ f o r l i s _ l o o m o n e x p e r _ u p _ w e l t i o n s </s> e^-98.5396345359351
<s> m a i d _ n o w s _ s e s i o n e s _ a r e a m e _ b e t _ t h e _ o f _ c h a s _ t h e _ t h a r e t i o n s t a k i n _ e z _ t h e s s _ f r i n g _ l o t h a t _ t o _ r e a r t h
e r s _ f o r _ a r _ e n t s _ m i n g _ u p _ b e e n o t _ h a d m i s _ t o u t </s> e^-244.914598004735
<s> s t _ c i s m e n c r o n _ i m p l a c k _ t h e _ a g e _ s a y s _ f o l v e _ t h i m p u - g h _ t h e _ t h e _ t h i s _ t h e _ b i e s _ i n g _ i s _ s t _ g o l d i s _ r a i
n d _ r o u t _ t h i n g _ j e c t u r _ d i s _ b e i t e u r o d _ s u r _ t h _ a _ m i l l _ b e f f k c q _ n e y </s> e^-301.856737868788
<s> i t h e _ i n g _ t i m p d v a t _ i n n i g _ i n g _ i s _ t h e _ t o o k _ q u e s _ h i n g _ t h e _ p a r e s t r u p _ l a n d o s s e s </s> e^-127.776220640781
<s> h e r i c t i o n e a t _ o n t e p q v - u _ h i s _ d o _ h a t _ s t _ b e t i o n i e r n i a l l _ a n d e n s _ d o l e _ t a _ b e c y c a t e d _ b e a r r i n g _ j u s h e _ a
n o v e r s e _ u p _ t v k y </s> e^-243.602117044793
<s> f o r m _ w i t y _ b e y _ t o _ i s t _ t o d _ m o u n t l y _ c l i c a l _ r e _ s p e o p l a s s _ i n _ w e _ i n g _ o v e _ b u t _ p r o m e n t _ h a t _ c a r e _ </s> e^-16
6.342097791436
<s> p a k e r _ h o m _ f o r _ a _ m o _ r a c h _ f o r k i n _ h o m _ t h e _ t h e _ e x p e n _ h a d e n _ s c a m p r i e s s _ i n g t _ g r e d _ r o f _ f i s e r v e s t i r o b
a n n i a _ f a c i s _ a n _ u p p e t h e n e _ s i t i l s o o m e s p o w n _ t h e _ c a u x _ x h _ s a i n g </s> e^-324.50501222884
<s> t h r o p e o n t h _ a g e d _ h e _ h i n s t _ u n o t _ t h </s> 1.91656359631813e-30
<s> t h e _ t h i s _ c a r t a c h i s </s> 1.49532860909238e-15
```

unigram language model depends on probability of each letter appearing in corpora; it's not able to construct any words correctly. But different picture can be seen for the sequences generated by the bigram and trigram model. We can see the bigram model can generate small words such as 'to', 'one', 'air' etc. And it also steps up its game of sequence generation because it knows which word generally appears after a particular word. Trigram has better performance than bigram. It gets a lot of words correct, such as, 'for', 'the', 'age', 'of', 'his' etc. and the incorrect words have bigger subset of letter sequences from the correct words.
So, these results produced by the language models totally makes sense.

2. Restoring vowels.
Just as in HW1, we can decode text without vowels. Try doing that experiment on test.txt with the language models you trained. What's your command line? What are your accuracy results? Include the input file test.txt.novowels and the result files test.txt.vowel_restored.{uni,bi,tri}. (Hint: you can use sed -e 's/[aeiou]//g' to remove vowels).

Answer:

First I removed the vowels from the test.txt using the following command.

```
        cat test.txt | sed -e 's/[aeiou]//g' > test.txt.novowels
```

Then I used the following command to restore the vowels. Here we have composed our language model with remove-vowel.fst [From HW1]

```
cat test.txt.novowels | sed -e 's/ /_/g;s/\(.\)/\1 /g' | awk
'{printf("<s> %s </s>\n", $0)}' | carmel -sribIEWk 1
<language_model>.wfsa remove-vowels.fst >
test.txt.vowel_restored.<language_model>
```

We tried to restore vowels using all three of our language models. And the accuracy generated from eval2.py are given below.

Language Model	Accuracy
Unigram	recall= 0.688 precision= 1.000 F1= 0.815
Bigram	recall= 0.836 precision= 0.996 F1= 0.909
Trigram	recall= 0.928 precision= 0.978 F1= 0.952

Now, we have modified our test.txt before evaluation into test.txt.modified. Because the test.txt.vowel_restored.<language_model> has spaces and starts and ends with special symbol <s> </s>. So, We used the following command to change the test file.

```
cat test.txt | sed -e 's/ /_/g;s/\(.\)/\1 /g' | awk '{printf("<s> %s
</s>\n", $0)}' > test.txt.modified
```

And then we performed evaluation using the following command:

```
python eval2.py test.txt.modified
test.txt.vowel_restored.<{uni/bi/tri}>
```

3. Restoring spaces.
Similarly, we can remove the spaces and try to restore them with the help of language models. Try doing that experiment on test.txt with the language models you trained. What's your command line? What are your accuracy results? Include the input file test.txt.nospaces and the result files test.txt.space_restored.{uni,bi,tri}. (Hint: you can use sed -e 's/ //g' to remove spaces).

Finally, decode the following two sentences with your models:
therestcanbeatotalmessandyoucanstillreaditwithoutaproblem
thisisbecausethehumanminddoesnotreadeveryletterbyitselfbutthewordasawhole.
Answer:

Similar to the approach for the above problem, first I removed the spaces from the test.txt using the following command.

```
        cat test.txt | sed -e 's/ //g' > test.txt.nospaces
```

Then I used the following command to restore the spaces. Here we have composed our language model with remove-spaces.fst

```
cat test.txt.nospaces | sed -e 's/\(.\)/\1 /g' | awk '{printf("<s> %s
</s>\n", $0)}' | carmel -sribIEWk 1 <language_model>.wfsa
remove-spaces.fst > test.txt.space_restored.<language_model>
```

We tried to restore vowels using all three of our language models. And the accuracy generated from eval2.py are given below.

Language Model	Accuracy
Unigram	recall= 0.842 precision= 1.000 F1= 0.914
Bigram	recall= 0.951 precision= 0.999 F1= 0.974
Trigram	recall= 0.993 precision= 0.992 F1= 0.993

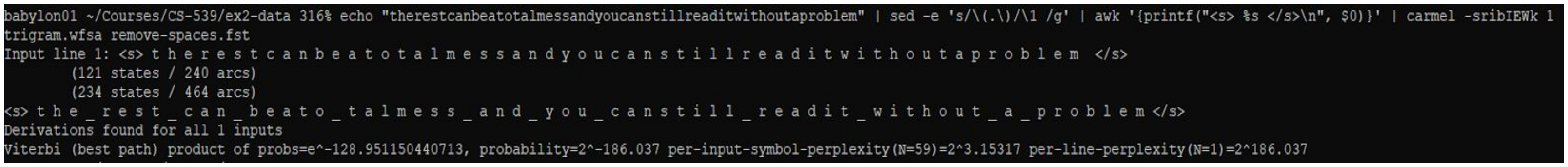
Now, we have modified our test.txt before evaluation into test.txt.modified. Because the test.txt.vowel_restored.<language_model> has spaces and starts and ends with special symbol <s> </s>. So, We used the following command to change the test file.

```
cat test.txt | sed -e 's/ /_/g;s/\(.\)/\1 /g' | awk '{printf("<s> %s
</s>\n", $0)}' > test.txt.modified
```

And then we performed evaluation using the following command:

```
python eval2.py test.txt.modified
test.txt.space_restored.<{uni/bi/tri}>
```

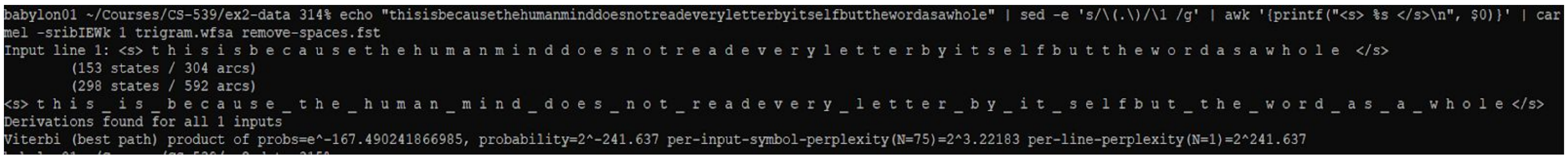
We decoded the two lines given in the question using the trigram language model. The input and



```
babylon01 ~/Courses/CS-539/ex2-data 316% echo "therestcanbeatotalmessandyoucanstillreaditwithoutaproblem" | sed -e 's/\(.\)/\1 /g' | awk '{printf("<s> %s </s>\n", $0)}' | carmel -sribIEWk 1
trigram.wfsa remove-spaces.fst
Input line 1: <s> t h e r e s t c a n b e a t o t a l m e s s a n d y o u c a n s t i l l r e a d i t w i t h o u t a p r o b l e m  </s>
        (121 states / 240 arcs)
        (234 states / 464 arcs)
<s> t h e _ r e s t _ c a n _ b e a t o _ t a l m e s s _ a n d _ y o u _ c a n s t i l l _ r e a d i t _ w i t h o u t _ a _ p r o b l e m </s>
Derivations found for all 1 inputs
Viterbi (best path) product of probs=e^-128.951150440713, probability=2^-186.037 per-input-symbol-perplexity(N=59)=2^3.15317 per-line-perplexity(N=1)=2^186.037
```

output are both given below:

<s> t h e _ r e s t _ c a n _ b e a t o _ t a l m e s s _ a n d _ y o u _ c a n s t i l l _ r e a d i t _ w i t h o u t _ a _ p r o b l e m </s>



```
babylon01 ~/Courses/CS-539/ex2-data 314% echo "thisisbecausethehumanminddoesnotreadeveryletterbyitselfbutthewordasawhole" | sed -e 's/\(.\)/\1 /g' | awk '{printf("<s> %s </s>\n", $0)}' | car
mel -sribIEWk 1 trigram.wfsa remove-spaces.fst
Input line 1: <s> t h i s i s b e c a u s e t h e h u m a n m i n d d o e s n o t r e a d e v e r y l e t t e r b y i t s e l f b u t t h e w o r d a s a w h o l e  </s>
        (153 states / 304 arcs)
        (298 states / 592 arcs)
<s> t h i s _ i s _ b e c a u s e _ t h e _ h u m a n _ m i n d _ d o e s _ n o t _ r e a d e v e r y _ l e t t e r _ b y _ i t _ s e l f b u t _ t h e _ w o r d _ a s _ a _ w h o l e </s>
Derivations found for all 1 inputs
Viterbi (best path) product of probs=e^-167.490241866985, probability=2^-241.637 per-input-symbol-perplexity(N=75)=2^3.22183 per-line-perplexity(N=1)=2^241.637
```

<s> t h i s _ i s _ b e c a u s e _ t h e _ h u m a n _ m i n d _ d o e s _ n o t _ r e a d e v e r y _ l e t t e r _ b y _ i t _ s e l f b u t _ t h e _ w o r d _ a s _ a _ w h o l e </s>

<u>Commands Used:</u>

```
echo
"thisisbecausethehumanminddoesnotreadeveryletterbyitselfbutthewordasa
whole" | sed -e 's/\(.\)/\1 /g' | awk '{printf("<s> %s </s>\n", $0)}'
| carmel -sribIEWk 1 trigram.wfsa remove-spaces.fst
```

4. Which of the two decoding problems is easier? Write a paragraph of observations you made in these experiments.
<u>Answer:</u>

I think there were some issues with the provided evaluation file; which doesn't reflect properly the accuracy of our language models in vowel restoration and space restoration tasks. For example unigram model performed horrendously in both restoration tasks; it wasn't able to make of the restorations right. But it is not reflected into the accuracies we got running the evaluation file. But trigram model showed the notion that we are moving in the correct path for both restoration tasks. For example, in the space restoration task the trigram model is correctly identifying some of the words.
Among the two decoding problems the space restoration problem is easier. Because in the vowel restoration problem there are lots of ambiguities even for an n-gram case. Using past n-gram it's really hard to predict which vowel will come next. For example, Where, What these 2 words starts with 'wh'; and in English language corpus there are lots of occurrences of what and where. It would be difficult for a trigram model to predict if the next vowel is e/a seeing 'WH'. Restoring spaces are a bit simpler than that. Because there are some letters after which it's easier to predict the word is coming to an end.