# Quiz Submissions - Quiz 2

**WENXIN ZHANG (username: wz2164)**

**Attempt 2**

Written: Oct 1, 2021 2:46 PM - Oct 1, 2021 2:50 PM

**Submission View**

Your quiz has been submitted successfully.

**Question 1** **1 / 1 point**

Suppose we are using a 4-gram language model. Would the conditional distribution of the next token be the same for these two inputs?

**Input 1: 'the' 'quick' 'brown' fox' 'jumps' 'over' 'the' <next token>**

**Input 2: 'the' 'quick' 'blue' fox' 'jumps' 'over' 'the' <next token>**

- Yes
- No
- Insufficient information to determine

**Question 2** **0 / 1 point**

Suppose we are using a 4-gram language model. Would the conditional distribution of the next token be the same for these two sentences?

**Input 1: 'the' 'quick' 'brown' fox' 'jumps' 'over' 'the' <next token>**

**Input 2: 'the' 'quick' 'brown' fox' 'leaps' 'over' 'the' <next token>**

- Yes
- No
- Insufficient information to determine

**Question 3** **0 / 1 point**

What are the potential benefits of using a larger N N-gram model? (Choose all that could apply.)

- [ ] More information is available in a larger context to inform the prediction
- [ ] More memory efficient to predict using a larger context
- [ ] The model sees a more diverse set of N-grams in context, making its predictions more robust

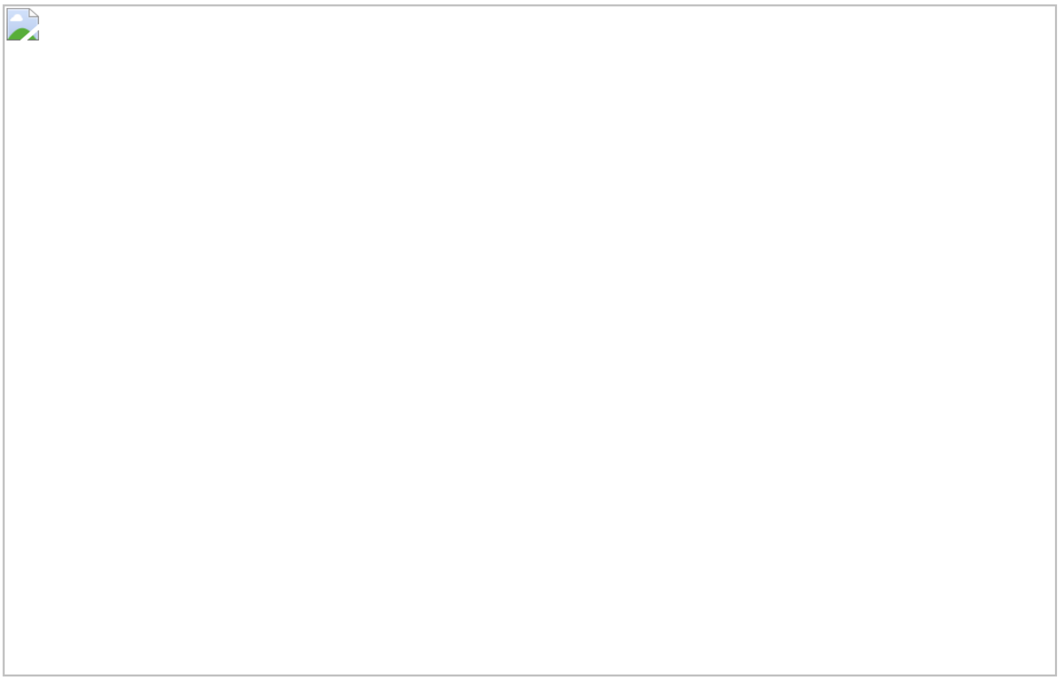
**Question 4** **1 / 1 point**

What are the potential downsides of using a larger N N-gram model? (Choose all that could apply)

- [ ] Many N-grams may only be seen a small number of times in training, which means the empirically learned probability distribution is built on a relatively small number of examples
- [ ] Many N-grams may never be seen during training, and 0 probability will be assigned to them
- [ ] Requires storing more N-grams

**Question 5** **1 / 1 point**

Remember the following code for our Ngram feed-forward neural network in Lab 3:



Is this implementation sensitive to the word order of the input N-gram prefix?

- ( ) Yes
- ( ) No

**Question 6** **1 / 1 point**

I want to train a language model (e.g. N-gram) that predicts text backwards. To do so, I carefully apply [::-1] or the .reverse method to the list of tokens (after tokenization), producing sequences like these:

**Input: '<eos>' '.' 'dog' 'my' 'like' 'I' '<bos>'**

I then train my model as usual. At inference time, I sample a new sentence by providing the following input:

**Input: '<eos>'**

Is this procedure likely to work?

- Yes
- No

### Question 7 — 1 / 1 point

I want to train a language model (e.g. N-gram) that predicts text backwards. To do so, I train my model as usual, but at inference time, I sample a new sentence by providing the following input:

**Input: '<eos>'**

Is this procedure likely to work?

- Yes
- No

### Question 8 — 1 / 1 point

What is the space complexity for storing a count-based N-gram model?
(If you are unfamiliar with what "space complexity" means, read it as "at what rate will the memory required to store the model grow, in the worst case scenario".)

- O(vocab * N)
- O(vocab ^ N)
- O(N ^ vocab)
- O(1)

### Question 9 — 1 / 1 point

Which of the following statements are true about feedforward N-gram models and count-based N-gram models? (Choose all that could apply)

- [ ] Feedforward N-gram models always perform better than N-gram models
- [ ] Count-based N-grams will assign 0 probability to unseen N-grams, but feedforward N-gram models may not
- [ ] Both models do not take into account tokens outside of the (N-1) token context
- [ ] Training either model requires you to load the whole dataset into memory at once

**Question 10** **1 / 1 point**

You have an n-gram distribution which you smooth with add-1 smoothing. The entropy of the smoothed distribution is __ the entropy of the original distribution.

- ○ smaller than
- ○ greater than or equal to

**Question 11** **1 / 1 point**

Why might a (well-trained) language model assign a lower probability to a grammatical sentence A as compared to a non-grammatical sentence B?

We should not only focus on the grammar but also the meaning conveyed by the sentence.

Considering that if grammatical sentence A seems more unlikely to happen based on its common sense than non-grammatical sentence B, a well-trained language model may still assign a lower probability to grammatical A.

Take an example here; sentence A is "Look! The rabbits are eating lions on the grass"; sentence B is "Look! The rabbits are eating carrot on the grass". Though there is a grammatical error in sentence B (the correct version is: eating carrots), it makes more sense than sentence A since we can never imagine rabbits eating lions!

Language model uses ML model to score the probability of the sentences, which will assign a higher conditional probability to the following words if the context is more related or the related counts are larger.

**Question 12** **1 / 1 point**

Consider a character-based language model (each token is a letter, punctuation mark, or space) and a word-based language model (each token is a word, or punctuation mark). If it helps, consider them both to be N-gram models, working on English.

Name one advantage of a character-based model over the word-based model, and one advantage of the word-based model over the character-based model.

The main advantage of character-based language models is that their discrete space is much smaller than the word-based model. There are nearly 97 English-language characters in common usage, including all punctuation marks. However, a vocabulary may consist of many thousands of words and quickly lead to a curse of dimensionality.

The main advantage of word-based language model is that it can distinguish the words with very similar character sequences but quite different meanings, such as "quiet" and "quite", due to the meaningful nature of words.

**Question 13** **0.1 / 0.1 points**

How long did you spend to complete this quiz (in hours)?

1.2 Hour.

Done