**Software Analytics - Assignment 9**

Due date: 11:59 pm Nov 26, 2023.

**General information**

You can use all available resources: notes, AI apps, Google search... But you need to do all the work on your own. You cannot discuss or share your work with your friends or classmates.

The instructor and TA will not provide any clarification or suggestions. You should work with the best of your knowledge and understanding.

You need to submit a document containing your answers and the R code you use to produce the analysis and answers. You should copy the analysis results and figures produced to your answer document. Please ensure that you submit both the Word/PDF solution file and the "Rhistory" file, which is necessary for a complete evaluation of your work.

Note: If you don't submit the Rhistory file, you will be deducted 50% of the total points.

For each question, before writing the code, you should explain your ideas on how to solve the problem: Why did you choose a particular method, function, or solution? What was your thought process in approaching the problem? Your descriptions will account for half of the points for each question. Providing clear and thoughtful explanations shows that you understand the topic and will be crucial for achieving full marks.

**Question**

Assume you want to build a language model for a simple File class having 6 methods: open, close, read, write, seek, eof. The codebase has the following usages of this class:

|  |
| --- |
| ST open seek read read write close ED |
| ST open read write read write read write close ED |
| ST open write write write close ED |
| ST open read close open write close ED |
| ST seek read write ED |
| ST open eof read close ED |
| ST open seek read seek read close ED |
| ST eof read ED |
| ST open seek write write write close ED |
| ST open seek read write seek read write close ED |

Q1 (1 pt). Including two special tokens ST and ED, what is the vocabulary V for this model?

We can easily determine the vocabulary of the model by counting the unique commands in each method, giving us the following result

Vocabulary : V = [ST, ED, open, close, read, write, seek, eof]

Q2 (1 pt). Estimate a unigram model M1 using that training data (i.e., calculate P(v) for each v in V).

For this unigram, we will be counting each occurrence of a method from our codebase, the, calculate a probability by dividing the count by the total methods in the corpus. For simplicity sake and consistency, we will be using up to four decimals.

|  |  |  |
| --- | --- | --- |
| Unigram | Count in corpus | Probability |
| ST | 10 | 10/74 or 0.1351 |
| Open | 9 | 9/74 or 0.1216 |
| Read | 13 | 13/74 or 0.1756 |
| Write | 14 | 14/74 or 0.1891 |
| Seek | 7 | 7/74 or 0.0945 |
| Eof | 2 | 2/74 or 0.0270 |
| close | 9 | 9/74 or 0.1216 |
| ED | 10 | 10/74 or 0.1351 |
| total | 74 |  |

Q3 (2 pt). Estimate a bigram model M2 using that training data (i.e., calculate P(v|u) for all pairs of u v).

For this bigram, we will be counting the method call pairs in our codeblock and calculate their probability by dividing the total pairs by the total amount of occurrences of each method

|  |
| --- |
| ST open seek read read write close ED |
| ST open read write read write read write close ED |
| ST open write write write close ED |
| ST open read close open write close ED |
| ST seek read write ED |
| ST open eof read close ED |
| ST open seek read seek read close ED |
| ST eof read ED |
| ST open seek write write write close ED |
| ST open seek read write seek read write close ED |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| u-v | ST | open | read | write | seek | eof | close | ED |
| ST | 0 | 8  P(open | St) = 8/10 | 0 | 0 | 1  P(seek | St) = 1/10 | 1  P(eof | St) = 1/10 | 0 | 0 |
| Open | 0 | 0 | 2  P(read | open) = 2/9 | 2  P(write | open) = 2/9 | 4  P(seek | open) = 4/9 | 1  P(eof | open) = 1/9 | 0 | 0 |
| Read | 0 | 0 | 1  P(read | read) = 1/13 | 7  P(write | read) = 7/13 | 1  P(seek | read) = 1/13 | 0 | 3  P(close | read) = 3/13 | 1  P(ED | read) = 1/13 |
| Write | 0 | 0 | 2  P(read | write) = 2/14 | 4  P(write | write) = 4/14 | 1  P(seek | write) = 1/14 | 0 | 6  P(close | write) = 6/14 | 1  P(ED | write) = 1/14 |
| Seek | 0 | 0 | 6  P(read | seek) = 6/7 | 1  P(write | seek) = 1/7 | 0 | 0 | 0 | 0 |
| Eof | 0 | 0 | 2  P(read | eof) = 2/2 | 0 | 0 | 0 | 0 | 0 |
| close | 0 | 1 P(open | close) = 1/9 | 0 | 0 | 0 | 0 | 0 | 8  P(ED | close) = 8/9 |
| ED | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Q4 (1 pt). Use model M1 to calculate the probability of two sequences:

We will be answering these questions by multiplying the probability of each individual method by each other in the order they appear in the sequence.

s1: open read close = Prob(open) \* Prob(read) \* Prob(close) = 0.1216 \* 0.1756 \* 0.1216 = 0.0025

s2: close read open = Prob(close) \* Prob(read) \* Prob(open) = 0.1216 \* 0.1756 \* 0.1216 = 0.0025

Do they have the same probability? Why? In practice, what sequence is more likely to appear? Why?

Prob(s1) = Prob(s2), both sequences have the same probability because the individual probabilities of close and open are the same, and even if they were different, their order would not have any effect in the result as we are considering their individual probabilities instead of pair probabilities. In practice, s1 is more likely to appear, because we follow a set of rules that indicates that a file must be opened before we can actually close it.

Q5 (1 pt). Use model M2 to calculate the probability of those two sequences. Do they still have the same probability? If not, why?

We will use a similar approach from the previous question, but this time we will include the pair probabilities of each method, in addition to adding the methods ST and ED.

S1 = ST open read close ED = P(open|ST) \* P(read|open) \* P(close|read) \* P(ED|close) = 8/10 \* 2/9 \*3/13\*8/9 = 0.0364

S2 = St close read open ED = P(close|ST) \* P(read|close) \* P(open|read) \* P(ED|open) = 0 \* 0 \* 0 \* 0 = 0

No, they have different probabilities, this is due because we are calculating the probability of a pair of methods appearing in a specific order. Is more noticeable when the order breaks the rules of the code block and results in in a probability of zero.

Q6 (2 pt). The specification for this class is to use **open** before **close**. Can two models M1 and M2 model that specification? Why? How about the specification that we need to open before calling read or write?

No, because the specification of using open before close can be far away from each other and we cannot determine with either M1 or M2 the accuracy of the model. Especially M2, there is not a pair for closing after opening consecutively, so it cannot learn the long dependency. While is possible for M2 to model the relation of opening before being able to read or write, it still far from an accurate model, what would happen if the first command after open is seek? The relation would be lost, and the model could not associate that long dependency. In short, M1 and M2 cannot model these specifications correctly because they are long dependencies that they cannot associate with.

Q7 (2 pt). Assume that the programmer has written "open write". What are two most likely tokens he will write next?

Either “open write close” or “open write write”

Q8. We want to build a model with longer context. So we decide to use linear regression model. First, we encode each token as numbers ST = 0; ED = 1; open = 0.1, close = 0.9, read = 0.3, write = 0.7, seek = 0.4, eof = 0.6.

Then we extract training data into input and output. We decide to use two tokens as context. So, each input contains two numbers for two prior tokens and the output is the number for the next token. For example:

|  |  |  |
| --- | --- | --- |
| Token sub-sequences | Input | Output |
| open read read | 0.1 0.3 | 0.3 |
| read read write | 0.3 0.3 | 0.7 |
| write close ED | 0.7 0.9 | 1 |

a (2 pt). Produce the input - output table for the whole training data. Note: you need to extract and encode all sub-sequences of size 3.

We will follow the instructions and construct the table as indicated, for convenience ill separate the input as X1 and X2

|  |  |  |  |
| --- | --- | --- | --- |
| Sub sequence | input | | output |
| X1 | X2 | Y1 |
| ST open seek | 0 | 0.1 | 0.4 |
| open seek read | 0.1 | 0.4 | 0.3 |
| seek read read | 0.4 | 0.3 | 0.3 |
| read read write | 0.3 | 0.3 | 0.7 |
| read write close | 0.3 | 0.7 | 0.9 |
| write close ED | 0.7 | 0.9 | 1 |
| ST open read | 0 | 0.1 | 0.3 |
| open read write | 0.1 | 0.3 | 0.7 |
| read write read | 0.3 | 0.7 | 0.3 |
| write read write | 0.7 | 0.3 | 0.7 |
| ST open write | 0 | 0.1 | 0.7 |
| open write write | 0.1 | 0.7 | 0.7 |
| write write write | 0.7 | 0.7 | 0.7 |
| write write close | 0.7 | 0.7 | 0.9 |
| open read close | 0.1 | 0.3 | 0.9 |
| read close open | 0.3 | 0.9 | 0.1 |
| close open write | 0.9 | 0.1 | 0.7 |
| open write close | 0.1 | 0.7 | 0.9 |
| ST seek read | 0 | 0.4 | 0.3 |
| seek read write | 0.4 | 0.3 | 0.7 |
| read write ED | 0.3 | 0.7 | 1 |
| ST open eof | 0 | 0.1 | 0.6 |
| open eof read | 0.1 | 0.6 | 0.3 |
| eof read close | 0.6 | 0.3 | 0.9 |
| read close ED | 0.3 | 0.9 | 1 |
| seek read seek | 0.4 | 0.3 | 0.4 |
| read seek read | 0.3 | 0.4 | 0.3 |
| seek read close | 0.4 | 0.3 | 0.9 |
| ST eof read | 0 | 0.6 | 0.3 |
| eof read ED | 0.6 | 0.3 | 1 |
| open seek write | 0.1 | 0.4 | 0.7 |
| seek write write | 0.4 | 0.7 | 0.7 |
| read write seek | 0.3 | 0.7 | 0.4 |
| write seek read | 0.7 | 0.4 | 0.3 |

b (2 pt). Build a linear regression model for such training data.

For this, we will load the data of our table in a data frame, such as data.frame(x1,x2,y), then use that data to construct a linear model using lm() with the formula y ~ x1 + x2 as indicated above:

> x1 = c(0, 0.1, 0.4, 0.3, 0.3, 0.7, 0, 0.1, 0.3, 0.7, 0, 0.1, 0.7, 0.7, 0.1, 0.3, 0.9, 0.1, 0, 0.4, 0.3, 0, 0.1, 0.6, 0.3, 0.4, 0.3, 0.4, 0, 0.6, 0.1, 0.4, 0.3, 0.7)

> x2 = c(0.1, 0.4, 0.3, 0.3, 0.7, 0.9, 0.1, 0.3, 0.7, 0.3, 0.1, 0.7, 0.7, 0.7, 0.3, 0.9, 0.1, 0.7, 0.4, 0.3, 0.7, 0.1, 0.6, 0.3, 0.9, 0.3, 0.4, 0.3, 0.6, 0.3, 0.4, 0.7, 0.7, 0.4)

> y = c(0.4, 0.3, 0.3, 0.7, 0.9, 1, 0.3, 0.7, 0.3, 0.7, 0.7, 0.7, 0.7, 0.9, 0.9, 0.1, 0.7, 0.9, 0.3, 0.7, 1, 0.6, 0.3, 0.9, 1, 0.4, 0.3, 0.9, 0.3, 1, 0.7, 0.7, 0.4, 0.3)

> tridata = data.frame(x1,x2,y)

> tridata

> m1 = lm(y ~ x1 + x2,tridata)

> m1

A white background with red dots

Description automatically generatedA white background with red and blue text

Description automatically generated

c (1 pt). Use that model to predict what is the mostly likely token for the sequence in Q7. Is this a good prediction?

For this we will using the predict command, with our linear model (m1) as the model and a data frame reflecting the inputs of “open write”, this case being 0.1 and 0.7.

> predict(m1, data.frame(x1 = 0.1, x2 = 0.7))

A red text on a white background

Description automatically generated

Our result indicates that the most likely token is “eof”, this is not at all a good prediction, our bigram indicates that the most likely tokens are close and write, and even in our table above we can see the case that inputs 0.1 and 0.7 would produce 0.9 which is “close”

Q9. Representing each token as a single number might not be good enough. We decide to represent each token as a vector of size 2: ST = [0, 0], ED = [1, 1], open = [0.1, 0.5], close = [0.9, 0.5], read = [0.5, 0.3]; write = [0.5, 0.7],

seek = [0.2, 0.2], eof = [0.5, 0.5].

The input is encoded by vectors of two context tokens. Output is encoded as vector for the next token. The training data will look like this:

|  |  |  |
| --- | --- | --- |
| Token sub-sequences | Input  x1 x2 x3 x4 | Output  y1 y2 |
| open read read | 0.1 0.5 0.5 0.3 | 0.5 0.3 |
| read read write | 0.5 0.3 0.5 0.3 | 0.5 0.7 |
| write close ED | 0.5 0.7 0.9 0.5 | 1 1 |

a (2 pt). Produce the input - output table for the whole training data. Note: you need to extract and encode all sub-sequences of size 3.

Same approach and method as before, this time we will be adding x3, x4, and y2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Tokens sub-Sequence | **Inputs** | | | | **Outputs** | |
| X1 | X2 | X3 | X4 | Y1 | Y2 |
| ST open seek | 0 | 0 | 0.1 | 0.5 | 0.2 | 0.2 |
| open seek read | 0.1 | 0.5 | 0.2 | 0.2 | 0.5 | 0.3 |
| seek read read | 0.2 | 0.2 | 0.5 | 0.3 | 0.5 | 0.3 |
| read read write | 0.5 | 0.3 | 0.5 | 0.3 | 0.5 | 0.7 |
| read write close | 0.5 | 0.3 | 0.5 | 0.7 | 0.9 | 0.5 |
| write close ED | 0.5 | 0.7 | 0.9 | 0.5 | 1 | 1 |
| ST open read | 0 | 0 | 0.1 | 0.5 | 0.5 | 0.3 |
| open read write | 0.1 | 0.5 | 0.5 | 0.3 | 0.5 | 0.7 |
| read write read | 0.5 | 0.3 | 0.5 | 0.7 | 0.5 | 0.3 |
| write read write | 0.5 | 0.7 | 0.5 | 0.3 | 0.5 | 0.7 |
| ST open write | 0 | 0 | 0.1 | 0.5 | 0.5 | 0.7 |
| open write write | 0.1 | 0.5 | 0.5 | 0.7 | 0.5 | 0.7 |
| write write write | 0.5 | 0.7 | 0.5 | 0.7 | 0.5 | 0.7 |
| write write close | 0.5 | 0.7 | 0.5 | 0.7 | 0.9 | 0.5 |
| open read close | 0.1 | 0.5 | 0.5 | 0.3 | 0.9 | 0.5 |
| read close open | 0.5 | 0.3 | 0.9 | 0.5 | 0.1 | 0.5 |
| close open write | 0.9 | 0.5 | 0.1 | 0.5 | 0.5 | 0.7 |
| open write close | 0.1 | 0.5 | 0.5 | 0.7 | 0.9 | 0.5 |
| ST seek read | 0 | 0 | 0.2 | 0.2 | 0.5 | 0.3 |
| seek read write | 0.2 | 0.2 | 0.5 | 0.3 | 0.5 | 0.7 |
| read write ED | 0.5 | 0.3 | 0.5 | 0.7 | 1 | 1 |
| ST open eof | 0 | 0 | 0.1 | 0.5 | 0.5 | 0.5 |
| open eof read | 0.1 | 0.5 | 0.5 | 0.5 | 0.5 | 0.3 |
| eof read close | 0.5 | 0.5 | 0.5 | 0.3 | 0.9 | 0.5 |
| read close ED | 0.5 | 0.3 | 0.9 | 0.5 | 1 | 1 |
| seek read seek | 0.2 | 0.2 | 0.5 | 0.3 | 0.2 | 0.2 |
| read seek read | 0.5 | 0.3 | 0.2 | 0.2 | 0.5 | 0.3 |
| seek read close | 0.2 | 0.2 | 0.5 | 0.3 | 0.9 | 0.5 |
| ST eof read | 0 | 0 | 0.5 | 0.5 | 0.5 | 0.3 |
| eof read ED | 0.5 | 0.5 | 0.5 | 0.3 | 1 | 1 |
| open seek write | 0.1 | 0.5 | 0.2 | 0.2 | 0.5 | 0.7 |
| seek write write | 0.2 | 0.2 | 0.5 | 0.7 | 0.5 | 0.7 |
| read write seek | 0.5 | 0.3 | 0.5 | 0.7 | 0.2 | 0.2 |
| write seek read | 0.5 | 0.7 | 0.2 | 0.2 | 0.5 | 0.3 |

b (2 pt). Build 2 logistic regression models to simulate a simple neural network model for such training data. (Note: you can use 2 formula y1 ~ x1 + x2 + x3 + x4 and y2 ~ x1 + x2 + x3 + x4).

Exact same process as before, this time we will making two models based on each output,

**> x1 = c(0, 0.1, 0.2, 0.5, 0.5, 0.5, 0, 0.1, 0.5, 0.5, 0, 0.1, 0.5, 0.5, 0.1, 0.5, 0.9, 0.1, 0, 0.2, 0.5, 0, 0.1, 0.5, 0.5, 0.2, 0.5, 0.2, 0, 0.5, 0.1, 0.2, 0.5, 0.5)**

**> x2 = c(0, 0.5, 0.2, 0.3, 0.3, 0.7, 0, 0.5, 0.3, 0.7, 0, 0.5, 0.7, 0.7, 0.5, 0.3, 0.5, 0.5, 0, 0.2, 0.3, 0, 0.5, 0.5, 0.3, 0.2, 0.3, 0.2, 0, 0.5, 0.5, 0.2, 0.3, 0.7)**

**> x3 = c(0.1, 0.2, 0.5, 0.5, 0.5, 0.9, 0.1, 0.5, 0.5, 0.5, 0.1, 0.5, 0.5, 0.5, 0.5, 0.9, 0.1, 0.5, 0.2, 0.5, 0.5, 0.1, 0.5, 0.5, 0.9, 0.5, 0.2, 0.5, 0.5, 0.5, 0.2, 0.5, 0.5, 0.2)**

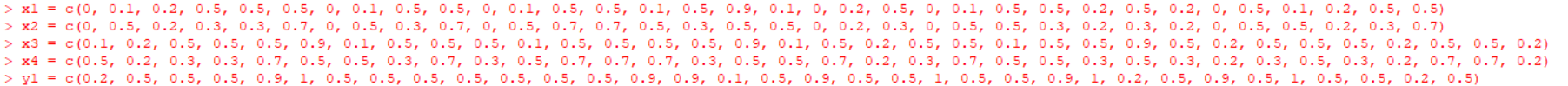
**> x4 = c(0.5, 0.2, 0.3, 0.3, 0.7, 0.5, 0.5, 0.3, 0.7, 0.3, 0.5, 0.7, 0.7, 0.7, 0.3, 0.5, 0.5, 0.7, 0.2, 0.3, 0.7, 0.5, 0.5, 0.3, 0.5, 0.3, 0.2, 0.3, 0.5, 0.3, 0.2, 0.7, 0.7, 0.2)**

**> y1 = c(0.2, 0.5, 0.5, 0.5, 0.9, 1, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.9, 0.9, 0.1, 0.5, 0.9, 0.5, 0.5, 1, 0.5, 0.5, 0.9, 1, 0.2, 0.5, 0.9, 0.5, 1, 0.5, 0.5, 0.2, 0.5)**

y2 = c(0.2, 0.3, 0.3, 0.7, 0.5, 1, 0.3, 0.7, 0.3, 0.7, 0.7, 0.7, 0.7, 0.5, 0.5, 0.5, 0.7, 0.5, 0.3, 0.7, 1, 0.5, 0.3, 0.5, 1, 0.2, 0.3, 0.5, 0.3, 1, 0.7, 0.7, 0.2, 0.3)

> m1 = lm(y1 ~ x1 + x2 + x3 + x4, sixgram)

> m2 = lm(y2 ~ x1 + x2 + x3 + x4, sixgram)

A white background with red text

Description automatically generated

c (1 pt). Use those models to predict what is the mostly likely token for the sequence in Q7. Is this a good prediction?

Same process as before, this time we will use the double input values to get each respective output. Then we will compare them to get the predicted token

> predict(m1, data.frame(x1 = 0.1, x2 = 0.5, x3 = 0.5, x4 = 0.7))

1

0.6655216

> predict(m2, data.frame(x1 = 0.1, x2 = 0.5, x3 = 0.5, x4 = 0.7))

1

0.5695594

A white sheet with red and blue text

Description automatically generated

We can predict that the next token [0.66, 0.56], for ease of analysis we will round up to the closest number which are [0.7, 0.6] is closer to the write and close tokens, which is what our bigram points out, however, there is still more room for improvement as its not giving an almost precise output. In short, is a better prediction compared to the previous model but not ideal yet.

Q10. Now we want to train the real neural model for this problem. Package **neuralnet** in R allows us to use symbols for output. Therefore, the training data can look like this:

|  |  |  |
| --- | --- | --- |
| Token sub-sequences | Input  x1 x2 x3 x4 | Output |
| open read read | 0.1 0.5 0.5 0.3 | read |
| read read write | 0.5 0.3 0.5 0.3 | write |
| write close ED | 0.5 0.7 0.9 0.5 | ED |

a (1 pt). Produce the input - output table for the whole training data. Note: you need to extract and encode all sub-sequences of size 3.

Same process, different output

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tokens sub-Sequence | **Inputs** | | | | **Output** |
| X1 | X2 | X3 | X4 |
| ST open seek | 0 | 0 | 0.1 | 0.5 | SEEK |
| open seek read | 0.1 | 0.5 | 0.2 | 0.2 | READ |
| seek read read | 0.2 | 0.2 | 0.5 | 0.3 | READ |
| read read write | 0.5 | 0.3 | 0.5 | 0.3 | WRITE |
| read write close | 0.5 | 0.3 | 0.5 | 0.7 | CLOSE |
| write close ED | 0.5 | 0.7 | 0.9 | 0.5 | ED |
| ST open read | 0 | 0 | 0.1 | 0.5 | READ |
| open read write | 0.1 | 0.5 | 0.5 | 0.3 | WRITE |
| read write read | 0.5 | 0.3 | 0.5 | 0.7 | READ |
| write read write | 0.5 | 0.7 | 0.5 | 0.3 | WRITE |
| ST open write | 0 | 0 | 0.1 | 0.5 | WRITE |
| open write write | 0.1 | 0.5 | 0.5 | 0.7 | WRITE |
| write write write | 0.5 | 0.7 | 0.5 | 0.7 | WRITE |
| write write close | 0.5 | 0.7 | 0.5 | 0.7 | CLOSE |
| open read close | 0.1 | 0.5 | 0.5 | 0.3 | CLOSE |
| read close open | 0.5 | 0.3 | 0.9 | 0.5 | OPEN |
| close open write | 0.9 | 0.5 | 0.1 | 0.5 | WRITE |
| open write close | 0.1 | 0.5 | 0.5 | 0.7 | CLOSE |
| ST seek read | 0 | 0 | 0.2 | 0.2 | READ |
| seek read write | 0.2 | 0.2 | 0.5 | 0.3 | WRITE |
| read write ED | 0.5 | 0.3 | 0.5 | 0.7 | ED |
| ST open eof | 0 | 0 | 0.1 | 0.5 | EOF |
| open eof read | 0.1 | 0.5 | 0.5 | 0.5 | READ |
| eof read close | 0.5 | 0.5 | 0.5 | 0.3 | CLOSE |
| read close ED | 0.5 | 0.3 | 0.9 | 0.5 | ED |
| seek read seek | 0.2 | 0.2 | 0.5 | 0.3 | SEEK |
| read seek read | 0.5 | 0.3 | 0.2 | 0.2 | READ |
| seek read close | 0.2 | 0.2 | 0.5 | 0.3 | CLOSE |
| ST eof read | 0 | 0 | 0.5 | 0.5 | READ |
| eof read ED | 0.5 | 0.5 | 0.5 | 0.3 | READ |
| open seek write | 0.1 | 0.5 | 0.2 | 0.2 | ED |
| seek write write | 0.2 | 0.2 | 0.5 | 0.7 | WRITE |
| read write seek | 0.5 | 0.3 | 0.5 | 0.7 | SEEK |
| write seek read | 0.5 | 0.7 | 0.2 | 0.2 | READ |

b (3 pt). Build a neural network model for such training data. Try at least 3 choices for the size of the hidden layer.

For this, it will take multiple steps: First, we will export our training data to excel to convert it into a csv file that R4.3.1 will read and store in a variable. Then we will install the necessary packages and libraries for neuralnet, after successfully installing the files, we will create a neuralnet variable using neuralnet(y ~ ., data = our cvs file).  
  
> NeuronData = read.csv("dataNeuron.csv")

> NeuronData

> library(neuralnet)

Warning message:

package ‘neuralnet’ was built under R version 4.3.2

> net = neuralnet(y ~., data = NeuronData)

> net

A screenshot of a computer

Description automatically generated

Now we will try 3 different models with different hidden layers to obtain a better prediction model by using the following command

> net = neuralnet(y ~., data = NeuronData, hidden = “number of layers”)

> net = neuralnet(y ~., data = NeuronData, hidden = 4)

> net2 = neuralnet(y ~., data = NeuronData, hidden = 6)

> plot(net)

> plot(net2)

> net3 = neuralnet(y ~., data = NeuronData, hidden = 10)

> plot(net3)

A diagram of a network

Description automatically generatedA diagram of a network

Description automatically generated

A diagram of a network

Description automatically generated

I cannot manage to make the error value appear clearly, but the initial impression is that it has a error value of 8.18.

c (1 pt). Use the best model to predict what is the mostly likely token for the sequence in Q7. Is this a good prediction? Is this better than the prediction in Q9?

Now that we have our models, we will use the predict equivalent of the neuralnet library, compute()in the following manner to predict the most likely token for the Q7 sequence. We will be using “net3” as it has the lowest value of error using the following command

> compute(net3, data.frame(x1 = 0.1, x2 = 0.5, x3 = 0.5, x4 = 0.7))

A white background with numbers and red text

Description automatically generated

As we can see, the token with highest probability is [,7] with a value of 0.55, this is EOF which is wrong and far from the approximate prediction from Q9, which at least had values closer to our predicted tokens in our bigram and training data.

Q11. We want to predict a token based on a token before it and a token after it. For example: if the programmer write code: "open [.] close" then it is likely that the token in the blank [.] will be "read" or "write". This model would be a masked language model.

a (1 pt). Why a masked language model is not considered a proper language model? Although is considered a language model, is not a fully independent one because is purpose is for pre-training a model; in other words, masked language objective is make the model understand the context and relations of the given data and produce results associated with that, even if the results are incorrect.

b (1 pt). Do you think that masked language model will provide better prediction for code than a bigram or trigram model? Why? Yes, while in the short term bigrams may provide a clearer picture, a well trained masked model will be able to predict a decently correct answer even if it has a long dependency, which the bigram and trigram will not be able to predict .

c (1 pt). Build the training data for a masked language model as in Q8 and Q9. The input output table will look like the following. Note: the output is the token in the middle, while the input contains vectors of the tokens before it and after it.

|  |  |  |
| --- | --- | --- |
| Token sub-sequences | Input  x1 x2 x3 x4 | Output |
| open read read | 0.1 0.5 0.5 0.3 | read |
| read read write | 0.5 0.3 0.5 0.7 | read |
| write close ED | 0.5 0.7 1 1 | close |

Same process as before, just exchanging the values of the outputs for the inputs.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Tokens sub-Sequence | **Inputs** | | | | **Output** |
| X1 | X2 | X3 | X4 |
| ST open seek | 0 | 0 | 0.2 | 0.2 | OPEN |
| open seek read | 0.1 | 0.5 | 0.5 | 0.3 | SEEK |
| seek read read | 0.2 | 0.2 | 0.5 | 0.3 | READ |
| read read write | 0.5 | 0.3 | 0.5 | 0.7 | READ |
| read write close | 0.5 | 0.3 | 0.9 | 0.5 | WRITE |
| write close ED | 0.5 | 0.7 | 1 | 1 | CLOSE |
| ST open read | 0 | 0 | 0.5 | 0.3 | OPEN |
| open read write | 0.1 | 0.5 | 0.5 | 0.7 | READ |
| read write read | 0.5 | 0.3 | 0.5 | 0.3 | WRITE |
| write read write | 0.5 | 0.7 | 0.5 | 0.7 | READ |
| ST open write | 0 | 0 | 0.5 | 0.7 | OPEN |
| open write write | 0.1 | 0.5 | 0.5 | 0.7 | WRITE |
| write write write | 0.5 | 0.7 | 0.5 | 0.7 | WRITE |
| write write close | 0.5 | 0.7 | 0.9 | 0.5 | WRITE |
| open read close | 0.1 | 0.5 | 0.9 | 0.5 | READ |
| read close open | 0.5 | 0.3 | 0.1 | 0.5 | CLOSE |
| close open write | 0.9 | 0.5 | 0.5 | 0.7 | OPEN |
| open write close | 0.1 | 0.5 | 0.9 | 0.5 | WRITE |
| ST seek read | 0 | 0 | 0.5 | 0.3 | SEEK |
| seek read write | 0.2 | 0.2 | 0.5 | 0.7 | READ |
| read write ED | 0.5 | 0.3 | 1 | 1 | WRITE |
| ST open eof | 0 | 0 | 0.5 | 0.5 | OPEN |
| open eof read | 0.1 | 0.5 | 0.5 | 0.3 | EOF |
| eof read close | 0.5 | 0.5 | 0.9 | 0.5 | READ |
| read close ED | 0.5 | 0.3 | 1 | 1 | CLOSE |
| seek read seek | 0.2 | 0.2 | 0.2 | 0.2 | READ |
| read seek read | 0.5 | 0.3 | 0.5 | 0.3 | SEEK |
| seek read close | 0.2 | 0.2 | 0.9 | 0.5 | READ |
| ST eof read | 0 | 0 | 0.5 | 0.3 | EOF |
| eof read ED | 0.5 | 0.5 | 1 | 1 | READ |
| open seek write | 0.1 | 0.5 | 0.5 | 0.7 | SEEK |
| seek write write | 0.2 | 0.2 | 0.5 | 0.7 | WRITE |
| read write seek | 0.5 | 0.3 | 0.2 | 0.2 | WRITE |
| write seek read | 0.5 | 0.7 | 0.5 | 0.3 | SEEK |

d (1 pt). Build a neural network model for such training data. Try at least 3 choices for the size of the hidden layer.

We will do the same process we did in the previous question; the only difference is how the data is organized. The output is the token in the middle and the input is the values of the last token.

> maskeddata = read.csv("maskedData.csv")

> masked1 = neuralnet(y ~., data = maskeddata, hidden = 4)

> masked2 = neuralnet(y ~., data = maskeddata, hidden = 8)

> masked3 = neuralnet(y ~., data = maskeddata, hidden = 12)

A white background with red text

Description automatically generated

A diagram of a network

Description automatically generated

A diagram of a network

Description automatically generated

A diagram of a network

Description automatically generated

As we notice that the more layers the net has, the smaller the error, so we are going to use the third model as our prediction tool

e (1 pt). Use the best model to predict what is the mostly likely token for the blank in "open [.] close". Is this a good prediction?

We will use the same process as the last question, but instead of using the input values of the write, we use the values of the token close

> compute(masked3, data.frame(x1 = 0.1, x2 = 0.5, x3 = 0.9, x4 = 0.5))

A white background with numbers and letters

Description automatically generated

As we can see, the probability aims for object 4 (0.4878) which is WRITE, so is a pretty good prediction of our table.