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:

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

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

To identify the vocabulary(V) of the given language model, we need to consider all unique tokens including the special tokens ST and ED.

The vocabulary(V) for this model is: {‘ST, ‘open’, ‘eof’, ’close’, ‘write’,’ ED’,’ read’,’ seek’}.

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

The ‘R’ code to estimate a unigram M1 model for a given language is as follows:

> training\_data <- c(

+ "open seek read read write close",

+ "open read write read write read write close",

+ "open write write write close",

+ "open read close open write close",

+ "seek read write",

+ "open eof read close",

+ "open seek read seek read close",

+ "eof read",

+ "open seek write write write close",

+ "open seek read write seek read write close"

+ )

> training\_data <- paste ("ST", training\_data, "ED", sep = " ")

> tokens <- unlist (strsplit (training\_data, split = " "))

> token\_counts <- table(tokens)

> unigram\_probs <- token\_counts / sum(token\_counts)

> print(unigram\_probs)

In the above line of code at first, we define the training\_data which stores the given language which has combinations of read, write, close, open, eof, and seek and we also include special tokens like ST and ED. Later we create a list of tokens by splitting the training\_data string using strsplit() function and by split = “ “ argument. The table () function is then used to count the occurrences of each token. The frequency of each token in the training\_data is shown in this table. Lastly, the algorithm divides the token counts by the total number of tokens to determine the unigram probabilities (unigram\_probs). The possibility that each token will appear in the model is represented by these probabilities.

Output:

tokens

close ED eof open read seek ST write

0.1216 0.1351 0.0270 0.1216 0.1756 0.0945 0.1351 0.1891

Note: The output decimals were rounded to 4 digits to paste the output in one line.

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

The ‘R’ code to estimate a bigram model M2 for the training data is as follows:

> deriveBigramProbabilities <- function(training\_data) {

+ training\_data <- paste("ST", training\_data, "ED", sep = " ")

+ tokens <- unlist(strsplit(training\_data, split = " "))

+ count\_Bigram <- matrix(0, nrow = length(unique(tokens)), ncol = length(unique(tokens)),

+ dimnames = list(unique(tokens), unique(tokens)))

+ for (i in 1:(length(tokens) - 1)) {

+ count\_Bigram[tokens[i], tokens[i + 1]] <- count\_Bigram[tokens[i], tokens[i + 1]] + 1

+ }

+ bigramProbabilities <- count\_Bigram / rowSums(count\_Bigram)

+ return(bigramProbabilities)

+ }

> training\_data <- c(

+ "open seek read read write close",

+ "open read write read write read write close",

+ "open write write write close",

+ "open read close open write close",

+ "seek read write",

+ "open eof read close",

+ "open seek read seek read close",

+ "eof read",

+ "open seek write write write close",

+ "open seek read write seek read write close"

+ )

> bigramProbabilities <- deriveBigramProbabilities(training\_data)

> print(bigramProbabilities, digits = 4)

In the above ‘R’ code a t first, we create a function named deriveBigramProbabilities where the training data is tokenized by adding special tokens ST and ED. Later we initialize a matrix called count\_Bigram to store the counts of Bigrams. The dimensions of this matrix are estimated by the unique tokens in a given model. Later we iterate a for loop, this loop iterates each consecutive pair of tokens presented in training data and updates the corresponding entry in the created matrix. Now we exit from the for loop and calculate the probability of the bigrams and we named a variable called bigramProbabilities to store the probabilities of the bigrams. Based on the frequency of successive token pair occurrences in a particular corpus, Bigram probabilities are computed. Mathematically, the probability of encountering token *j* given that the previous token was *i* is calculated as = .

In this code, we compute the bigram probability by dividing each count by the total counts in the associated row. This guarantees that for every row, the probabilities add up to 1.

The results are stored and printed in bigramProbabilities.

Output:

ST open seek read write close ED eof

ST 0 0.8000 0.10000 0.00000 0.0000 0.0000 0.00000 0.1000

open 0 0.0000 0.44444 0.22222 0.2222 0.0000 0.00000 0.1111

seek 0 0.0000 0.00000 0.85714 0.1429 0.0000 0.00000 0.0000

read 0 0.0000 0.07692 0.07692 0.5385 0.2308 0.07692 0.0000

write 0 0.0000 0.07143 0.14286 0.2857 0.4286 0.07143 0.0000

close 0 0.1111 0.00000 0.00000 0.0000 0.0000 0.88889 0.0000

ED 1 0.0000 0.00000 0.00000 0.0000 0.0000 0.00000 0.0000

eof 0 0.0000 0.00000 1.00000 0.0000 0.0000 0.00000 0.0000

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

s1: open read close

s2: close read open

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

By considering the Model M1 we have the following data :

tokens

close ED eof open read seek ST write

0.1216 0.1351 0.0270 0.1216 0.1756 0.0945 0.1351 0.1891

Where the numerical value represents the corresponding unigram probabilities of the respective tokens.

Let us consider the two sequences.

s1: open read close

s2: close read open

Now we derive the sequences as :

Sequences s1: open read close

*P*(*s*1)=*P*(open)×*P*(read)×*P*(close)

Sequence s2: close read open

*P*(*s*2)=*P*(close)×*P*(read)×*P*(open)

Now by substituting the values:

For *P*(*s*1)

*P*(*s*1) = 0.1216×0.1756×0.1216

For *P*(*s*2)

*P*(*s*2) = 0.1216×0.1756×0.1216

The probability ratio for both sequences will be the same since this unigram model ignores the relationships between subsequent tokens and just considers the individual probabilities of each token. In conclusion, the two sequences have same likely chance to appear. In actuality, the sequential dependencies in the data might be better captured by a bigram or higher-order model.

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

By considering the model M2 we have the following data:

ST open seek read write close ED eof

ST 0 0.8000 0.10000 0.00000 0.0000 0.0000 0.00000 0.1000

open 0 0.0000 0.44444 0.22222 0.2222 0.0000 0.00000 0.1111

seek 0 0.0000 0.00000 0.85714 0.1429 0.0000 0.00000 0.0000

read 0 0.0000 0.07692 0.07692 0.5385 0.2308 0.07692 0.0000

write 0 0.0000 0.07143 0.14286 0.2857 0.4286 0.07143 0.0000

close 0 0.1111 0.00000 0.00000 0.0000 0.0000 0.88889 0.0000

ED 1 0.0000 0.00000 0.00000 0.0000 0.0000 0.00000 0.0000

eof 0 0.0000 0.00000 1.00000 0.0000 0.0000 0.00000 0.0000

Where the numerical value represents the corresponding bigram probabilities of the respective tokens.

Let us consider the two sequences.

s1: open read close

s2: close read open

Now we derive the sequences as :

Sequences s1: open read close

*P*(*s*1)=*P*(read/open)×*P*(close/read)

Sequence s2: close read open

*P*(*s*2)=*P*(read/close)×*P*(open/read)

Now by substituting the values:

For *P*(*s*1)

*P*(*s*1) = 0.2222 × 0.2308 = 0.05123

For *P*(*s*2)

*P*(*s*2) = 0 × 0 = 0

So, by the above results, we can state that the probabilities of both sequences are different.

By considering the sequence s1: open read close

This sequence has a non-zero probability which states that this sequence is likely to appear by considering the theoretical derivation of the probability from observed frequencies of the model M2.

By considering the sequence s2: close read open

This sequence has zero probability, and it also states that this sequence is not observed highly and not likely to appear by considering the theoretical derivation of the probability from observed frequencies of the model M2.

Q6. 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?

Specification: Use "open" before "close"

• Model M1: Sequential relationships between tokens are not directly captured by the unigram model M1. It offers probability for every token separately. Consequently, M1 is unable to formally enforce the order of "open" before "close."

• Model M2: Tokens' conditional probabilities given their previous tokens are represented by the bigram model M2, which may be able to inferentially capture the sequence "open" before "close." In the observed data used to build the model, "close" is assumed to emerge after "open" if the probability of "close" given "open" is non-zero. Consequently, in M2, a non-zero probability of "close" given "open" might imply specification compliance.

Specification: Open before calling Read or Write

• Model M1: M1 lacks the ability to automatically collect sequential patterns, much as the "open" before "close" requirements. Each token is given a probability, and the precise order of "open" followed by "read" or "write" is not encoded.

• Model M2: According to this specification, the bigram model M2 might be able to recognize the sequence "open" before "read" or "write." According to the observed data that was used to create the model, "read" or "write" tends to follow "open" if the probability of "read" or "write" given "open" are non-zero. Therefore, non-zero probabilities of "write" or "read" after "open" in M2 may indicate compliance with this standard.

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

To mention the two most likely tokens that the programmer writes next, first, we consider the Bigram model to predict the tokens. The ‘R’ code to do this is as follows:

> bigram\_M2\_model <- matrix(

+ c(

+ 0, 0.8000, 0.10000, 0.00000, 0.0000, 0.0000, 0.00000, 0.1000,

+ 0, 0.0000, 0.44444, 0.22222, 0.2222, 0.0000, 0.00000, 0.1111,

+ 0, 0.0000, 0.00000, 0.85714, 0.1429, 0.0000, 0.00000, 0.0000,

+ 0, 0.0000, 0.07692, 0.07692, 0.5385, 0.2308, 0.07692, 0.0000,

+ 0, 0.0000, 0.07143, 0.14286, 0.2857, 0.4286, 0.07143, 0.0000,

+ 0, 0.1111, 0.00000, 0.00000, 0.0000, 0.0000, 0.88889, 0.0000,

+ 1, 0.0000, 0.00000, 0.00000, 0.0000, 0.0000, 0.00000, 0.0000,

+ 0, 0.0000, 0.00000, 1.00000, 0.0000, 0.0000, 0.00000, 0.0000

+ ),

+ nrow = 8, byrow = TRUE,

+ dimnames = list(

+ c("ST", "open", "seek", "read", "write", "close", "ED", "eof"),

+ c("ST", "open", "seek", "read", "write", "close", "ED", "eof")

+ )

+ )

> present\_tokens <- c("open", "write")

> probabilities <- bigram\_M2\_model[, present\_tokens]

> row\_index <- match (present\_tokens [1], colnames(bigram\_M2\_model))

> probabilities <- bigram\_M2\_model[row\_index,]

> output\_most\_likely\_tokens <- names (sort (probabilities, decreasing = TRUE)[1:2])

> cat("Two most likely tokens following 'open write':", output\_most\_likely\_tokens[1], "and", output\_most\_likely\_tokens[2], "\n")

In the above code at first, we generate the bigram\_M2\_model which is a matrix that represents the transition probabilities between the tokens of the bigram model. Later we specify the variable named present\_tokens which already contains the given tokens in a sequence such as “open write”. Later, For the given current tokens, the bigram\_M2\_model's probability matrix is extracted. It includes the odds of switching from each current token to every potential token that could come after. The code determines the row index (row\_index) of the present tokens ("open") row that corresponds to the first token. The initial token ("open") corresponds to a row from which probabilities are extracted. By a similar procedure, we operate the second token in the list of present tokens to estimate most likely tokens. Sorting the probability in descending order and choosing the top two yields the two most likely tokens after "open write".

Output:

Two most likely tokens following 'open write': seek and read

We can also estimate the two most likely tokens for the given sequence “open write” by considering unigram model M1 as follows:

The probabilities are dependent on how frequently each token appears in the training data because the unigram model ignores dependencies between characters. Among the potential tokens in this scenario, "Write" has the highest individual probability with a value of (14/74).

So by considering unigram model M1 the most likely token to appear for the given sequence are

“open write” : is write write.

Q8. 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) Produce the input-output table for the whole training data. Note: you need to extract and encode all sub-sequences of size 3.

|  |  |  |
| --- | --- | --- |
| Token sub-sequences | Input | Output |
| 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 |
| read write read | 0.3 0.7 | 0.3 |
| write read write | 0.7 0.3 | 0.7 |
| read write close | 0.3 0.7 | 0.9 |
| write close ED | 0.7 0.9 | 1 |
| 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 |
| write close ED | 0.7 0.9 | 1 |
| ST open read | 0 0.1 | 0.3 |
| 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 |
| write close ED | 0.7 0.9 | 1 |
| 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 |
| ST open seek | 0 0.1 | 0.4 |
| open seek read | 0.1 0.4 | 0.3 |
| 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 |
| read close ED | 0.3 0.9 | 1 |
| ST eof read | 0 0.6 | 0.3 |
| eof read ED | 0.6 0.3 | 1 |
| ST open seek | 0 0.1 | 0.4 |
| open seek write | 0.1 0.4 | 0.7 |
| seek write write | 0.4 0.7 | 0.7 |
| write write write | 0.7 0.7 | 0.7 |
| write write close | 0.7 0.7 | 0.9 |
| write close ED | 0.7 0.9 | 1 |
| ST open seek | 0 0.1 | 0.4 |
| open seek read | 0.1 0.4 | 0.3 |
| seek read write | 0.4 0.3 | 0.7 |
| read write seek | 0.3 0.7 | 0.4 |
| write seek read | 0.7 0.4 | 0.3 |
| seek read write | 0.4 0.3 | 0.7 |
| read write close | 0.3 0.7 | 0.9 |
| write close ED | 0.7 0.9 | 1 |

In the above table each input contains two numbers for two prior tokens and the output is the number for the next token.

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

The ‘R’ code to build a linear regression model for above training data is as follows:

> data8 <- read.csv("question8.csv")

> lm\_model\_M8<- lm(y ~ x1 + x2, data = data8)

> summary(lm\_model\_M8)

Here data8 is a variable that store the above generated training data.

Output:

Call:

lm(formula = y ~ x1 + x2, data = data8)

Residuals:

Min 1Q Median 3Q Max

-0.63310 -0.17051 0.04968 0.19158 0.40638

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 0.37388 0.06808 5.492 1.27e-06 \*\*\*

x1 0.42224 0.13090 3.226 0.0022 \*\*

x2 0.25838 0.12769 2.024 0.0483 \*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.2333 on 51 degrees of freedom

Multiple R-squared: 0.3218, Adjusted R-squared: 0.2952

F-statistic: 12.1 on 2 and 51 DF, p-value: 5.002e-05

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

From the above generated linear regression model to predict the most likely token after the sequence “ open write” , we need to run the below ‘R’ code as follows:

> predict(lm\_model\_M8, data.frame(x1=0.1, x2=0.7))

Output:

1

0.5969776

We got the encoded value as 0.5979 which is close to 0.6. From the given token eof has an encode value of 0.6. From this model we predict the most likely to appear token after “open write” sequence is eof. We can state that it is not the best fit prediction for categorical tokens but can be used for numerical tokens.

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

|  |  |  |
| --- | --- | --- |
| Token sub-sequences | Input  x1 x2 x3 x4 | Output  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 | |
| 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 | |
| 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 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 | |
| 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 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 | |
| write close ED | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.7 | 0.9 | 0.5 | | |  |  | | --- | --- | | 1 | 1 | |
| 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 | |
| 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 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 | |
| read close ED | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.3 | 0.9 | 0.5 | | |  |  | | --- | --- | | 1 | 1 | |
| 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 | |
| ST open seek | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0.1 | 0.5 | | |  |  | | --- | --- | | 0.2 | 0.2 | |
| 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 | |
| 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 | |
| write close ED | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.7 | 0.9 | 0.5 | | |  |  | | --- | --- | | 1 | 1 | |
| 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 write | |  |  |  |  | | --- | --- | --- | --- | | 0.2 | 0.2 | 0.5 | 0.3 | | |  |  | | --- | --- | | 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 | |
| seek read write | |  |  |  |  | | --- | --- | --- | --- | | 0.2 | 0.2 | 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 | |

In the above table each input is encoded by vectors of two context tokens. Output is encoded as vector for the next token.

b) 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).

The ‘R’ code to build logistic regression models is a s follows:

> data9 <- read.csv("question9.csv")

> y1\_model <- multinom(y1 ~ x1+x2+x3+x4, data= data9)

> summary(y1\_model)

Output of 1st logistic regression model:

multinom(formula = y1 ~ x1 + x2 + x3 + x4, data = data9)

Coefficients:

(Intercept) x1 x2 x3 x4

0.2 28.5460630 -8.872218 8.55629 -54.92269 12.128589

0.5 30.1367035 -11.450928 17.51931 -54.36370 9.176590

0.9 18.5975534 -10.235590 18.16848 -36.16904 11.432333

1 0.4015601 11.844231 13.28903 -14.34042 6.646039

Std. Errors:

(Intercept) x1 x2 x3 x4

0.2 36.78856 75.99112 23.50194 80.39371 123.7388

0.5 36.74317 75.81057 21.94673 80.31981 123.6897

0.9 33.02042 75.83202 21.97026 73.53089 123.6988

1 36.38276 86.36149 21.18057 67.92271 123.7249

Residual Deviance: 70.6322

AIC: 110.6322

The ‘R’ code to build the second logistic regression model is as follows:

> data9 <- read.csv("question9.csv")

> y2\_model <- multinom(y2 ~ x1 + x2 + x3 + x4, data = data9)

> summary(y2\_model)

Output:

Call:

multinom(formula = y2 ~ x1 + x2 + x3 + x4, data = data9)

Coefficients:

(Intercept) x1 x2 x3 x4

0.3 1.9014347 -2.922909 8.496179 -0.7537598 -3.9487833

0.5 -2.8903607 -1.968908 8.533480 5.4979515 -0.1039943

0.7 -0.3206616 -2.094372 10.102047 2.1868727 -2.8722166

1 -11.4546910 7.291306 10.084526 13.7424599 -3.7455703

Std. Errors:

(Intercept) x1 x2 x3 x4

0.3 2.113138 5.784976 8.878399 3.781412 4.146084

0.5 2.503551 5.855219 8.885000 4.454448 4.236609

0.7 2.192133 5.737213 8.851710 3.925219 4.130306

1 5.730576 9.834902 9.366772 5.062655 5.472430

Residual Deviance: 114.087

AIC: 154.087

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

The ‘R’ code to predict the sequence open write using the above two models is as follows:

> present\_data <- data.frame(x1 = 0.1, x2 = 0.5, x3 = 0.5, x4 = 0.7)

>

> prob\_y1 <- predict(y1\_model, newdata=present\_data, type="probs")

> prob\_y2 <- predict(y2\_model, newdata=present\_data, type="probs")

> print(prob\_y1)

0.1 0.2 0.5 0.9 1

2.485832e-08 1.062635e-02 5.962791e-01 3.930870e-01 7.522916e-06

> print(prob\_y2)

0.2 0.3 0.5 0.7 1

0.009964198 0.150686797 0.470909941 0.366063983 0.002375081

> y1\_most\_likely <- names(prob\_y1)[which.max(prob\_y1)]

> y2\_most\_likely <- names(prob\_y2)[which.max(prob\_y2)]

> print(y1\_most\_likely)

[1] "0.5"

> print(y2\_most\_likely)

[1] "0.5"

So from the above output by considering the values of y1 and y2 which are 0.5 we state that the token to most likely appear of the sequence “open write “is eof. It might not be the best choice for predicting vector-valued outputs or handling sequences

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

|  |  |  |
| --- | --- | --- |
| Token sub-sequences | Input  x1 x2 x3 x4 | Output |
| 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 |
| read write read | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.3 | 0.5 | 0.7 | | read |
| write read write | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.7 | 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 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 |
| write close ED | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.7 | 0.9 | 0.5 | | ED |
| ST open read | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0.1 | 0.5 | | read |
| 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 |
| write close ED | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.7 | 0.9 | 0.5 | | ED |
| 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 |
| ST open seek | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0.1 | 0.5 | | seek |
| open seek read | |  |  |  |  | | --- | --- | --- | --- | | 0.1 | 0.5 | 0.2 | 0.2 | | read |
| 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 |
| read close ED | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.3 | 0.9 | 0.5 | | ED |
| ST eof read | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0.5 | 0.5 | | read |
| eof read ED | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.5 | 0.5 | 0.3 | | ED |
| ST open seek | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0.1 | 0.5 | | seek |
| open seek write | |  |  |  |  | | --- | --- | --- | --- | | 0.1 | 0.5 | 0.2 | 0.2 | | write |
| seek write write | |  |  |  |  | | --- | --- | --- | --- | | 0.2 | 0.2 | 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 |
| write close ED | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.7 | 0.9 | 0.5 | | ED |
| ST open seek | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0.1 | 0.5 | | seek |
| open seek read | |  |  |  |  | | --- | --- | --- | --- | | 0.1 | 0.5 | 0.2 | 0.2 | | read |
| seek read write | |  |  |  |  | | --- | --- | --- | --- | | 0.2 | 0.2 | 0.5 | 0.3 | | 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 |
| seek read write | |  |  |  |  | | --- | --- | --- | --- | | 0.2 | 0.2 | 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 |

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

The ‘R’ code to build the first neural network model for above training data is as follows:

> data10 <- read.csv("question10.csv")

> net1 <- neuralnet(y ~ ., data = data10, hidden = 1)

> plot(net1)

Output:

A screenshot of a computer

Description automatically generated

To generate the second neural network we have following ‘R’ code

> net2 <- neuralnet(y ~ ., data = data10, hidden = 2)

> plot(net2)

Output:

A screenshot of a computer

Description automatically generated

The ‘R’ code to build a third neural network is as follows:

> net3 <- neuralnet(y ~ ., data = data10, hidden = 3)

> plot(net3)

Output:

A screenshot of a computer

Description automatically generated

c) 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?

By considering the neural-network 3 which has less errors than other neural networks below is the ‘R’ code to predict the sequence “open write”

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

Output:

$neurons

$neurons[[1]]

x1 x2 x3 x4

[1,] 1 0.1 0.5 0.5 0.7

$neurons[[2]]

[,1] [,2] [,3] [,4]

[1,] 1 0.2084474 3.228979e-07 0.03122049

$net.result

[,1] [,2] [,3] [,4] [,5] [,6] [,7]

[1,] 0.3192642 -0.01056115 0.007153641 -0.008775382 0.1405875 0.08737938 0.4643094

Among the neural\_network4.result , [7] 0.4643094 which is “ED” is the most likely next token.

In the given context, neural networks can be implemented best fit because they approach deep learning models, and can also compute complex patterns rather than logistic regression.

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) Why a masked language model is not considered a proper language model?

In natural language processing, masked language models (MLMs) are useful for representation learning. However, MLMs do not have an explicit joint distribution over language, which affects activities like generation and evaluation, in contrast to traditional models. Although MLMs are excellent at capturing local context, they oversimplify language complexity by assuming conditional independence across masked tokens. For jobs requiring a thorough comprehension of language, explicitly defining a joint distribution becomes essential, which presents difficulties when evaluating MLMs using common metrics such as confusion. One important factor in improving the performance of masked language models is to strike a balance between capturing complex language connections and facilitating efficient learning.

b) Do you think that masked language model will provide better prediction for code than a bigram or trigram model? Why?

The Unigram, Bigram, and Trigram models are examples of basic statistical techniques that analyze word probabilities in increasingly complicated sequences. While Bigram analyzes a word's probability given its predecessor, Trigram expands this to take into account two previous words, Unigram concentrates on the probabilities of individual words. These models, however basic, provide important information about word relationships within a sequence. On the other hand, pre-trained models designed for certain tasks like sentiment analysis are Masked Language Models (MLMs), which are based on the transformer architecture. With self-attention layers, MLMs are better than classic n-gram models at capturing complex relationships and contextual subtleties, which makes them better for prediction tasks, especially in the code domain.

c) 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 |

Required Table:

|  |  |  |
| --- | --- | --- |
| Token sub-sequences | Input  x1 x2 x3 x4 | Output |
| 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 |
| read write read | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.3 | 0.5 | 0.3 | | write |
| write read write | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.7 | 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 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 |
| write close ED | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.7 | 1 | 1 | | close |
| ST open read | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0.5 | 0.3 | | open |
| 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 |
| write close ED | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.7 | 1 | 1 | | close |
| 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 |
| ST open seek | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0.2 | 0.2 | | open |
| open seek read | |  |  |  |  | | --- | --- | --- | --- | | 0.1 | 0.5 | 0.5 | 0.3 | | seek |
| 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 |
| read close ED | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.3 | 1 | 1 | | close |
| ST eof read | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0.5 | 0.3 | | eof |
| eof read ED | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.5 | 1 | 1 | | read |
| ST open seek | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0.2 | 0.2 | | open |
| open seek write | |  |  |  |  | | --- | --- | --- | --- | | 0.1 | 0.5 | 0.5 | 0.7 | | seek |
| seek write write | |  |  |  |  | | --- | --- | --- | --- | | 0.2 | 0.2 | 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 |
| write close ED | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.7 | 1 | 1 | | close |
| ST open seek | |  |  |  |  | | --- | --- | --- | --- | | 0 | 0 | 0.2 | 0.2 | | open |
| open seek read | |  |  |  |  | | --- | --- | --- | --- | | 0.1 | 0.5 | 0.5 | 0.3 | | seek |
| seek read write | |  |  |  |  | | --- | --- | --- | --- | | 0.2 | 0.2 | 0.5 | 0.7 | | read |
| read write seek | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.3 | 0.2 | 0.2 | | write |
| write seek read | |  |  |  |  | | --- | --- | --- | --- | | 0.5 | 0.7 | 0.5 | 0.3 | | seek |
| seek read write | |  |  |  |  | | --- | --- | --- | --- | | 0.2 | 0.2 | 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 |

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

The ‘R’ code to build first neural network

> data11 <- read.csv("question11.csv")

> net1 = neuralnet(y~., data=data11, hidden=1)

> plot(net1)

Output:

A screenshot of a computer

Description automatically generated

The ‘R’ code to build second neural network

> net2 = neuralnet(y~., data=data11, hidden=2)

> plot(net2)

Output:

A screenshot of a computer

Description automatically generated

The ‘R’ code to build third neural network

> net3 = neuralnet(y~., data=data11, hidden=3)

> plot(net3)

Output:

A screenshot of a computer

Description automatically generated

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

The ‘R’ code to predict the most likely token for the blank in "open [.] close" by considering above third neural network which has least errors than other neural networks is as follows:

>compute(net3,data.frame(x1=0.1,x2=0.5,x3=0.9,x4=0.5))

Output:

$neurons

$neurons[[1]]

x1 x2 x3 x4

[1,] 1 0.1 0.5 0.9 0.5

$neurons[[2]]

[,1] [,2] [,3] [,4]

[1,] 1 0 0.9993624 0.0006023423

$net.result

[,1] [,2] [,3] [,4] [,5] [,6]

[1,] 0.01515886 0.01245913 0.02135667 0.4141615 0.09147274 0.4480387

By considering the highest value and by referring to the corresponding y column of the highest value from neural network third diagram we can state that

The most likely token for the blank in “open[.]close” is “eof”. This prediction employed a neural network visualization approach, suggesting a potential for a well-fitted model, enhancing the accuracy of predictions.

Note:

<https://web.stanford.edu/~jurafsky/slp3/> : Referred this website to learn and understand Unigram, Bigram, Tigram models.

<https://cran.r-project.org/web/packages/quanteda/quanteda.pdf> : Referred this website to learn and understand sub sequence tokens.

<https://www.r-bloggers.com/2015/09/how-to-perform-a-logistic-regression-in-r/>: Referred this website to learn and understand the regression models in R.

<https://rviews.rstudio.com/2020/07/20/shallow-neural-net-from-scratch-using-r-part-1/>: Referred this website to learn and understand the neural networks in R.