BIOS Homework 10

Homework 10

Student: Jama Brookes

Course Notes Language Models: https://github.com/rjenki/BIOS512/tree/main/lecture17
Unix: https://github.com/rjenki/BIOS512/tree/main/lecture18
Docker: https://github.com/rjenki/BIOS512/tree/main/lecture19

Question 1

```
# required libraries
library(httr)
library(tokenizers)
```

Make a language model that uses ngrams and allows the user to specify start words, but uses a random start if one is not specified.

```
tokenize_text <- function(text) {
    tokenizers::tokenize_words(text, lowercase=TRUE, strip_punct=TRUE)[[1]]
}
```

a) Make a function to tokenize the text.

```
key_from <- function(ngram, sep = "\x1f") {
    paste(ngram, collapse=sep)
}
```

b) Make a function generate keys for ngrams.

```
build_ngram_table <- function(tokens, n, sep = "\x1f") {
    if (length(tokens) < n) return(new.env(parent = emptyenv()))
    tbl <- new.env(parent = emptyenv())
```

```
    for (i in seq_len(length(tokens) - n + 1L)) {
        ngram <- tokens[i:(i + n - 2L)]
        next_word <- tokens[i + n - 1L]
        key <- paste(ngram, collapse = sep)
        counts <- if (!is.null(tbl[[key]])) tbl[[key]] else integer(0)
        if (next_word %in% names(counts)) {
            counts[[next_word]] <- counts[[next_word]] + 1L
        } else {
            counts[[next_word]] <- 1L
        }
        tbl[[key]] <- counts
    }
    tbl
}
```

c) Make a function to build an ngram table.

```
digest_text <- function(text, n) {
    tokens <- tokenize_text(text)
    build_ngram_table(tokens, n)
}
```

d) Function to digest the text.

```
digest_url <- function(url, n) {
    res <- httr::GET(url)
    txt <- httr::content(res, as = "text", encoding = "UTF-8")
    digest_text(txt,n)
}
```

e) Function to digest the url.

```
random_start <- function(tbl, sep = "\x1f") {
    keys <- ls(envir = tbl, all.names=TRUE)
    if (length(keys)==0) stop("No n-grams available. Digest text first.")
    picked <- sample(keys, 1)
    strsplit(picked, sep, fixed=TRUE)[[1]]
}
```

f) Function that gives random start.

```
predict_next_word <- function(tbl, ngram, sep = "\x1f") {
    key <- paste(ngram, collapse = sep)
    counts <- if(!is.null(tbl[[key]])) tbl[[key]] else integer(0)
    if (length(counts) == 0) return(NA_character_)
    sample(names(counts), size=1, prob=as.numeric(counts))
}
```

g) Function to predict the next word.

```
make_ngram_generator <- function(tbl, n, sep = "\x1f") {
    force(tbl); n <- as.integer(n); force(sep)
    function(start_words = NULL, length = 10L) {
        if ((is.null(start_words)) || length(start_words) != n - 1L) {
            start_words <- random_start(tbl, sep=sep)
        }
        word_sequence <- start_words
        for (i in seq_len(max(0L, length - length(start_words)))) {
            ngram <- tail(word_sequence, n - 1L)
            next_word <- predict_next_word(tbl, ngram, sep=sep)
            if (is.na(next_word)) break
            word_sequence <- c(word_sequence, next_word)
        }
        paste(word_sequence, collapse= " ")
    }
}
```

h) Function that puts everything together. Specify that if the user does not give a start word, then the random start will be used.

Question 2

For this question, set `seed=2025`.

```
set.seed(2025)
url <- "https://www.gutenberg.org/cache/epub/2591/pg2591.txt" # Grimm's Fairy Tails
tbl3 <- digest_url(url, n=3)
gen3 <- make_ngram_generator(tbl3, n=3)
```

a) Test your model using a text file of Grimm's Fairy Tails

```
output <- gen3(start_words = c("the", "king"), length = 15)
cat(output)
```

i) Using n=3, with the start word(s) "the king", with length=15.

```
## the king has forbidden me to marry another husband am not i shall ride upon
```

```
output <- gen3(length = 15)
cat(output)
```

ii) Using n=3, with no start word, with length=15.

```
## song was over the lake and herself into her little daughter's hand and was about
```

```
set.seed(2025)
url <- "https://www.gutenberg.org/cache/epub/46342/pg46342.txt" #Ancient Armour & Weapons in Europe
tbl3 <- digest_url(url, n=3)
gen4 <- make_ngram_generator(tbl3, n=3)
```

b) Test your model using a text file of Ancient Armour and Weapons in Europe

```
set.seed(2025)
output <- gen4(start_words = c("the", "king"), length = 15)
cat(output)
```

i) Using n=3, with the start word(s) "the king", with length=15.

```
## the king he added to the entire exclusion of the swords were made prisoners the
```

```
set.seed(2025)
output <- gen4(length = 15)
cat(output)
```

ii) Using n=3, with no start word, with length=15.

```
## lamentation de lemburn came forth completely armed after the fashion of this may be seen
```

c) Explain in 1-2 sentences the difference in content generated from each source. In the Grimm's Fairy Tails, the content has more of a ficitonal plot from a story, which is whimisical in nature. The Ancient Armour & Weapons in Europe produces more a more direct story that likely it talking about historical events and weaponry, like swords.

Question 3

a) What is a language learning model? A language model simply predicts the next word by using a probability distribution.

b) Imagine the internet goes down and you can't run to your favorite language model for help. How do you run one locally? We could run a language model locally by training one on information from public domain (like the ones on the internet today). We could do this by downloading OLLAMA, which uses language model tools, and then train it ourselves.

Question 4

Explain what the following vocab words mean in the context of typing `mkdir project` into the command line. If the term doesn't apply to this command, give the definition and/or an example.

Term	Meaning
Shell	The shell takes the input `mkdir project` and parses out what system to run/use.
Terminal emulator	Is the actual interface that you type mkdir project into.
Process	actually runs the code by creating a new "process" or running program.
Signal	we send these to tell the program to do something.
Standard input	reads the characters from the input.
Standard output	outputs characters if needed (typically no output is produce with mkdir commands)
Command line argument	is the thing we add to tell the process what to do. For the example, `project`
The environment	all the stuff a process can see when its running.

Question 5

Consider the following command `find . -iname "*.R" | xargs grep read_csv`.

a) What are the programs? find, xargs, and grep

b) Explain what this command is doing, part by part. `find .` - searches for the file in the current directory `-iname` - (with name of) ".R" (aka R files) `|` - pipe `xargs` - takes the files found `grep read_csv` - searchs for texts in the files and reads them

Question 6

Install Docker on your machine. See here for instructions.

a) Show the response when you run `docker run hello-world`. Hello from Docker! This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps: 1. The Docker client contacted the Docker daemon. 2. The Docker daemon pulled the "hello-world" image from Docker Hub. 3. The Docker daemon created a new container from that image. 4. The Docker daemon streamed that output to the Docker client.

Server is ready

b) Access Rstudio through a Docker container. Set your password and make sure your files show up on the Rstudio server. Type the command and the output you get below. docker run –platform linux/amd64 -it -e PASSWORD=dockertrial -p 8787:8787 rocker/verse

Adding password for user rstudio useradd: warning: the home directory already exists. Not copying any file from skel directory into it. rsession: no process found

RStudio Server Info

Version: 2024.04.0+492 Address: 0.0.0.0 Port: 8787 Username: rstudio Password: dockertrial

RStudio Server is ready to accept connections — open your browser and go to http://localhost:8787

c) How do you log in to the RStudio server? I went to the local host link and logged in with the username and password above.