Homework 10

Alexis Bryant

2025-11-20

Question 1

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.

a) Make a function to tokenize the text.

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

b) Make a function generate keys for Ngrams.

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

c) Make a function to build an Ngram table.

```
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
}
```

d) Function to digest the text.

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

e) Function to digest the url.

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

f) Function that gives random start.

```
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]]
}
```

g) Function to predict the next word.

```
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))
}
```

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

```
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 = " ")
  }
}
```

Question 2

For this question, set seed=2025.

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

```
set.seed(2025)

url <- "https://www.gutenberg.org/cache/epub/2591/pg2591.txt"
tbl3 <- digest_url(url, n = 3)
gen3 <- generator(tbl3, n = 3)
```

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

```
cat(gen3(start_words = c("the", "king"), length = 15), "\n\n")
```

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

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

```
cat(gen3(length = 15), "\n")
```

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

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

```
set.seed(2025)

url_2 <- "https://www.gutenberg.org/cache/epub/46342/pg46342.txt"
tbl3_2 <- digest_url(url_2, n = 3)
gen3_2 <- generator(tbl3_2, n = 3)
```

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

```
cat(gen3_2(start_words = c("the", "king"), length = 15), "\n\n")
```

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

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

```
cat(gen3_2(length = 15), "\n")
```

```
## king was campaigning in france denmark germany switzerland and livonia figures 5 and the sword
```

c) Explain in 1-2 sentences the difference in content generated from each source.

The Grimm's Fairy Tales model generated more storytelling-like language with human characters, emotions, and royal references. The Ancient Armour model generated more technical and descriptive language about objects and materials. The difference reflects the training corpus context: one narrative vs. one historical/technical.

Question 3

a) What is a language learning model?

A language model is a probabilistic model that predicts the next word in a sequence based on the context of previous words. It learns from patterns in large text to generate or understand language.

b) Imagine the internet goes down and you can't run to your favorite language model for help. How do you run one locally?

You can run a language model locally using Ollama, which wraps Docker-style containers to host models like Gemma or Llama on your own machine. After installing Ollama and pulling a model (like ollama pull gemma3:1b), run ollama serve to start a local API at localhost:11434. Then, in R, use httr and jsonlite to send POST requests to that local endpoint. This reproduces the same behavior as online APIs but without any network access.

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.

Shell:The program that interprets commands. It parses mkdir project and tells the system to make a directory.

Terminal emulator:The window or app (like VS Code terminal) that displays your shell.

Process:A running instance of a command mkdir becomes a short-lived process that interacts with the filesystem.

Signal: A message sent to control processes. Not applicable directly to mkdir.

Standard input (stdin): Data fed into a process (keyboard/file). mkdir doesn't read input, but cat < file.txt does.

Standard output (stdout): Text or data sent from a process. For mkdir, success prints nothing.

Command line argument: Additional text after the command. In mkdir project, project is an argument.

The environment: A set of system variables available to the shell. These determine where commands like mkdir are found.

Question 5

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

a) What are the programs?

find, xargs, grep,

b) Explain what this command is doing, part by part.

find . :

Searches recursively from the current directory (.).

-iname :

means case-insensitive match.

“*.R” :

Finds all files ending in .R.

|:

The pipe takes the list of files and passes it to the next command.

xargs grep read_csv :

For each file name output by find, runs grep read_csv inside it.So it searches all .R files for the string “read_csv”.The result lists where in your code you used read_csv()

Question 6

Install Docker on your machine.

a) Show the response when you run docker run hello-world.

- **I already have podman downloaded on my computer so I used podman instead **

When you run

```
podman run hello-world
```

you would get:

```
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.
(amd64) 3.
The Docker daemon created a new container from that image which runs the executable that produces the ou
4.
The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with: \$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID: <https://hub.docker.com/>

For more examples and ideas, visit: <https://docs.docker.com/get-started/>
```

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.

Command:

podman run -d -p 8787:8787 -e PASSWORD=rstudio -v $(pwd):/home/rstudio/project: Z bios512-hw .

Output:

[s6-init] making user provided files available at /var/run/s6/etc... exited 0. [s6-init] ensuring user provided files have correct perms... exited 0. [fix-attrs.d] applying ownership & permissions fixes... [fix-attrs.d] done. [cont-init.d] executing container initialization scripts... [cont-init.d] 01_set_env: executing... skipping /var/run/s6/container_environment/HOME skipping /var/run/s6/container_environment/PASSWORD skipping /var/run/s6/container_environment/RSTUDIO_VERSION [cont-init.d] 01_set_env: exited 0. [cont-init.d] 02_userconf: executing... [cont-init.d] 02_userconf: exited 0. [cont-init.d] done. [services.d] starting services [services.d] done.

c) How do you log in to the RStudio server?

Open your browser to http://localhost:8787

Username: rstudio Password: rstudio

You'll see your mounted folder (project/) and can knit or run your .Rmd files directly.