Homework #12

Sang Doan

11/20/2020

Folders: 1. `raw_data`: Original Winston Churchill's speeches. (existing folder)
2. `transcripts`: Transcripts of orginial speeches.
3. `json_files`: JSON files to feed to Amazon Transcribe.
4. `translated_texts`: French translations translated from English transcripts.
5. `translated_speeches`: French speeches synthesized from French translations.
6. `dynamic`: Dynamic files `french.txt`, `status.txt`, & `transcription_output.json`.

```
# Create folders
system('mkdir transcripts json_files translated_texts translated_speeches dynamic')
```

Note: The first three problems build towards problem 4.
Step 1. `submit_mp3.transcription()` uploads all Churchill speeches from folder `raw_data` to S3. Returns `<file names>`.
Step 2. `start_transcibing.transcription( <file names> )` transcribes all speeches. Returns `<job names>`.
Step 3. `check_status.transcription( <job names> )` is used repeatedly to check whether transcription jobs are done. Returns a logical vector.
Step 4. `get_transcript.transcription( <job names> )` returns `<transcripts>` in a character vector.
Step 5. `write_transcripts.transcription( <transcripts> )` writes transcripts to files in folder `transcripts`. Returns `<paths to transcripts>`.
Step 6. `translate( <paths to transcripts> )` returns `<French translations>` translated from English in a vector.
Step 7. `write_translations( <French translations> )` writes translated texts to files in folder `translated_texts`. Returns `<paths to French translations>`.
Step 8. `speak( <paths to French translations> )` converts French texts into mp3 files and saves them to folder `translated_speeches`. Returns `<paths to translated speeches>`.

```
namesSpeech <- list.files('raw_data') %>% tools::file_path_sans_ext() # Save file names without extensi
pathsSpeech <- list.files('raw_data') # Paths to all speeches

namesSpeech # These names are used for transcripts, translations, and translated speeches.
```

```
## [1] "1934-11-16_BBC_Winston_Churchill_The_Threat_Of_Nazi_Germany"
## [2] "1940-05-28_BBC_Winston_Churchill_On_Capitulation_Of_Belgium"
## [3] "1940-06-04_BBC_Winston_Churchill_We_Shall_Never_Surrender"
## [4] "1940-08-20_BBC_Winston_Churchill_The_First_Year"
## [5] "1941-06-16_BBC_Winston_Churchill_Broadcast_To_America"
## [6] "1941-08-29_BBC_Winston_Churchill_These_Are_Great_Days"
## [7] "1942-11-10_BBC_Winston_Churchill_The_End_Of_The_Beginning"
## [8] "1944-11-23_BBC_Winston_Churchill_Americas_Thanksgiving_Day"
```

Problem 1 (Steps 1, 2, 3, 4, & 5)

```
transcription_jobs <- pathsSpeech %>%
  # Step 1
  submit_mp3.transcription %>%
  # Step 2
  start_transcibing.transcription
```

```
# Step 3
repeat {
  completed <- check_status.transcription(transcription_jobs)
  if(all(completed)) break
}

# Step 4
transcripts <- get_transcript.transcription(transcription_jobs) %>%
  # Step 5
  write_transcripts.transcription(namesSpeech, .)
```

Now, transcript files are in folder transcripts.

```
transcripts
```

```
## [1] "transcripts/1934-11-16_BBC_Winston_Churchill_The_Threat_Of_Nazi_Germany.txt"
## [2] "transcripts/1940-05-28_BBC_Winston_Churchill_On_Capitulation_Of_Belgium.txt"
## [3] "transcripts/1940-06-04_BBC_Winston_Churchill_We_Shall_Never_Surrender.txt"
## [4] "transcripts/1940-08-20_BBC_Winston_Churchill_The_First_Year.txt"
## [5] "transcripts/1941-06-16_BBC_Winston_Churchill_Broadcast_To_America.txt"
## [6] "transcripts/1941-08-29_BBC_Winston_Churchill_These_Are_Great_Days.txt"
## [7] "transcripts/1942-11-10_BBC_Winston_Churchill_The_End_Of_The_Beginning.txt"
## [8] "transcripts/1944-11-23_BBC_Winston_Churchill_Americas_Thanksgiving_Day.txt"
```

Problem 2 (Steps 6 & 7)

```
translations <- transcripts %>%
  # Step 6
  translate %>%
  # Step 7
  write_translations(namesSpeech, .)

translations
```

```
## [1] "translated_texts/1934-11-16_BBC_Winston_Churchill_The_Threat_Of_Nazi_Germany.txt"
## [2] "translated_texts/1940-05-28_BBC_Winston_Churchill_On_Capitulation_Of_Belgium.txt"
## [3] "translated_texts/1940-06-04_BBC_Winston_Churchill_We_Shall_Never_Surrender.txt"
## [4] "translated_texts/1940-08-20_BBC_Winston_Churchill_The_First_Year.txt"
## [5] "translated_texts/1941-06-16_BBC_Winston_Churchill_Broadcast_To_America.txt"
## [6] "translated_texts/1941-08-29_BBC_Winston_Churchill_These_Are_Great_Days.txt"
## [7] "translated_texts/1942-11-10_BBC_Winston_Churchill_The_End_Of_The_Beginning.txt"
## [8] "translated_texts/1944-11-23_BBC_Winston_Churchill_Americas_Thanksgiving_Day.txt"
```

Problem 3 (Step 8)

```
translated_speeches <- translations %>% speak(namesSpeech, .)
```

Problem 4

```
list.files('transcripts')
```

```
## [1] "1934-11-16_BBC_Winston_Churchill_The_Threat_Of_Nazi_Germany.txt"
## [2] "1940-05-28_BBC_Winston_Churchill_On_Capitulation_Of_Belgium.txt"
## [3] "1940-06-04_BBC_Winston_Churchill_We_Shall_Never_Surrender.txt"
## [4] "1940-08-20_BBC_Winston_Churchill_The_First_Year.txt"
## [5] "1941-06-16_BBC_Winston_Churchill_Broadcast_To_America.txt"
## [6] "1941-08-29_BBC_Winston_Churchill_These_Are_Great_Days.txt"
## [7] "1942-11-10_BBC_Winston_Churchill_The_End_Of_The_Beginning.txt"
## [8] "1944-11-23_BBC_Winston_Churchill_Americas_Thanksgiving_Day.txt"
```

```
list.files('translated_speeches')
```

```
## [1] "1934-11-16_BBC_Winston_Churchill_The_Threat_Of_Nazi_Germany.mp3"
## [2] "1940-05-28_BBC_Winston_Churchill_On_Capitulation_Of_Belgium.mp3"
## [3] "1940-06-04_BBC_Winston_Churchill_We_Shall_Never_Surrender.mp3"
## [4] "1940-08-20_BBC_Winston_Churchill_The_First_Year.mp3"
## [5] "1941-06-16_BBC_Winston_Churchill_Broadcast_To_America.mp3"
## [6] "1941-08-29_BBC_Winston_Churchill_These_Are_Great_Days.mp3"
## [7] "1942-11-10_BBC_Winston_Churchill_The_End_Of_The_Beginning.mp3"
## [8] "1944-11-23_BBC_Winston_Churchill_Americas_Thanksgiving_Day.mp3"
```

Code

analysis.R

```
submit_mp3.transcription <- function(...) {
  # ... = file paths (various number of files allowed)
  # Returns file names as shown in S3

  files <- list(...)

  commands <- unlist(files) %>%
    paste('aws s3 cp raw_data/', ., ' s3://sang.math110', sep = '')
  for(i in commands) system(i)

  return(unlist(files))
}

start_transcibing.transcription <- function(...) {
  # ... = mp3 files stored on S3
  # Returns a vector of job names (equal to number of input files)

  file_names <- list(...)
  jobs <- vector(mode = 'character')
```

```
  # Create json files and transcribe
  for(i in unlist(file_names)) {
    job_name <- str_glue('math110', runif(1))
    jobs <- append(jobs, job_name)

    # Prepare json file
    out <- list(
      TranscriptionJobName = job_name,
      LanguageCode = 'en-US',
      MediaFormat = 'mp3',
      Media = list(MediaFileUri = str_glue('https://sang.math110.s3.amazonaws.com/', i)),
      OutputBucketName = 'sang.math110.transcribed'
    )

    # Write json file
    write_json(out, str_glue('json_files/', job_name, '.json'), auto_unbox = TRUE)

    # System call
    str_glue('aws transcribe start-transcription-job --region us-east-1 --cli-input-json file://json_fi
             job_name, '.json') %>%
      system
  }
  return(jobs)
}

check_status.transcription <- function(...) {
  # ... = job names to check
  # Returns a logical vector with length = number of input jobs
  # (TRUE = completed, FALSE = in progress or otherwise)

  out <- vector(mode = 'logical')
  jobs <- list(...)

  for(i in unlist(jobs)) {
    # 1. Identify the job & write its status to a txt file
    paste('aws transcribe list-transcription-jobs --region us-east-1 --job-name-contains',
          i, '> dynamic/status.txt') %>%
      system
    status <- read_json('dynamic/status.txt', simplifyVector = TRUE)

    # 2. Check if status is empty
    if(is_empty(status$TranscriptionJobSummaries$TranscriptionJobStatus)) {
      paste('aws transcribe list-transcription-jobs --region us-east-1 --job-name-contains',
            i, '--next-token', status$NextToken,'> dynamic/status.txt') %>%
        system
      status <- read_json('dynamic/status.txt', simplifyVector = TRUE)
    }

    # 3. Save status to out
    if(status$TranscriptionJobSummaries$TranscriptionJobStatus == 'COMPLETED')
      out <- append(out, TRUE)
    else out <- append(out, FALSE)
  }
```

```
  return(out)
}

get_transcript.transcription <- function(...) {
  # ... = job names
  # Returns a character vector of transcripts

  out <- vector(mode = 'character')
  jobs <- list(...)

  for(i in unlist(jobs)) {

    # Download transcript
    str_glue('aws s3 cp s3://sang.math110.transcribed/',
             i ,'.json dynamic/transcription_output.json') %>%
      system

    # Save transcript to out
    j <- read_json('dynamic/transcription_output.json', simplifyVector = TRUE)
    out <- j$results$transcripts$transcript %>%
      str_remove('\'|\"') %>%
      append(out, .)
  }

  return(out)
}

write_transcripts.transcription <- function(names, ...) {
  # names = names of text files
  # ... = transcripts (strings)
  # Returns a vector of paths to transcripts

  transcripts = list(...)
  i <- 1

  for(transcript in unlist(transcripts)) {
    writeLines(transcript, str_glue('transcripts/', names[i], '.txt'))
    i = i + 1
  }

  return(list.files('transcripts', full.names = TRUE))
}

translate <- function(...) {
  # ... = paths to original texts
  # Returns a character vector of translated texts

  files <- list(...)
  transcripts <- map_chr(unlist(files), function(x) readChar(x, nchars = 5000, useBytes = TRUE))
    # Read in at max 5000 bytes
  out <- vector(mode = 'character')

  for(transcript in transcripts) {
```

```
    # Translate
    Sys.setenv(text = transcript)
    system('aws translate translate-text --text \"$text\" --source-language-code en --target-language-c

    # Save translation to out
    j <- read_json('dynamic/french.txt', simplifyVector = TRUE)
    out <- j$TranslatedText %>%
      str_remove('\'|"') %>%
      append(out, .)
  }

  return(out)
}

write_translations <- function(names, ...) {
  # names = names of translated text files
  # ... = translations (strings)
  # Returns a vector of paths to French translations

  translations = list(...)
  i <- 1

  for(translation in unlist(translations)) {
    writeLines(translation, str_glue('translated_texts/', names[i], '.txt'))
    i = i + 1
  }

  return(list.files('translated_texts', full.names = TRUE))
}

speak <- function(names, ...) {
  # names = names of mp3 files
  # ... = text files
  # Returns a vector of paths to mp3 files

  textfiles <- list(...)
  i <- 1

  for(textfile in unlist(textfiles)) {

    # Convert text -> audio and store the audio file in folder translated_speeches
    Sys.setenv(text = readChar(textfile, nchar = 3000))
    str_glue('aws polly synthesize-speech --output-format mp3 --voice-id Mathieu --text \"$text\" transl
          names[i], '.mp3') %>%
      system

    i = i + 1
  }

  return(list.files('translated_speeches', full.names = TRUE))
}
```

config.R

```
source('analysis.R')

library(tidyverse)
library(magrittr)
library(httr)
library(jsonlite)
```