Reproducibility-Driven Development of a Medical Text Simplification Tool

Presented by - Ankit Pal
Deep Target NLP Research Group
Supervisor: Dr. Gondy Leroy

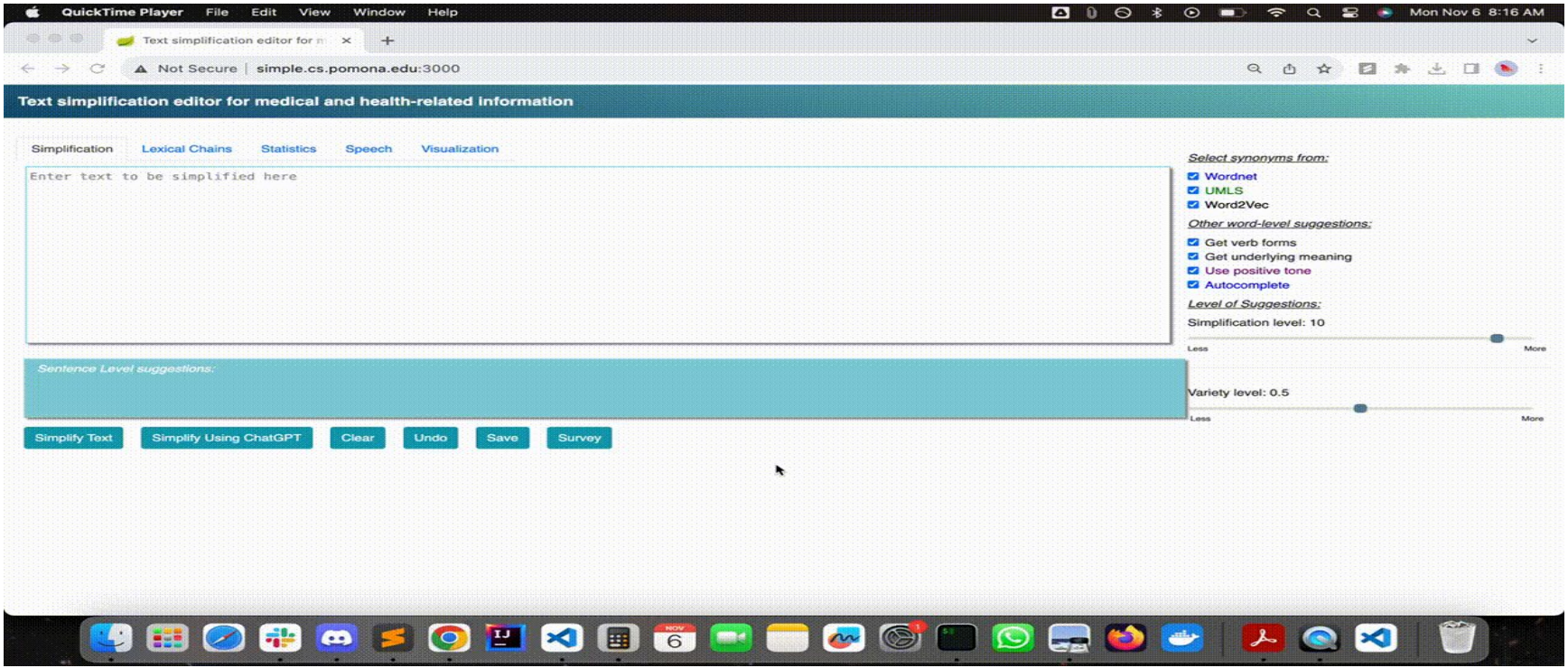
Why we do Medical Text Simplification?

- Limited health literacy is a barrier to accessing and understanding the health information that can empower patients, consumers and caregivers to monitor and manage their health, translate information into actionable and healthy behaviors, support decision-making, and guide healthcare consumption.
- Simplifying text can reduce this barrier and, potentially, reduce known disparities in health information.
- Few tools exist to support writing simplified text with demonstrated impact on comprehension.

What we do for medical text simplification?

- Development of Text Simplification Tool
- Conduct Research Studies

Text Simplification Tool



How we develop the tool and research study's content?

- Web Scraping
  - We use web scraping to collect medical data(texts)
- NLP Text Metrics
  - After collecting the data, we generate the NLP statistics of the texts
  - We collect 34 metrics from the texts for e.g. Average Grammar Frequency, Content Word Count, Nouns/Verbs/Adverbs/Adjectives Percentage, Number Of Lexical Chains, etc.
- Question Generation
  - For the studies, we generate different type of questions and answers using python
  - We used GPT-3.5 model and Spacy to generate questions and answers from the text
- Audio Generation and Scores
  - For the audio studies, we generate the audio from these texts
  - We use Azure speech service to generate the speech from the text and also to generate the speech score of the audio

How to make code reproducible?

- Github Repository
- Package Version Management
- Containerization
- Documentation
- Data Management

Github Repository

- We maintain all of the scripts in the [github repository]
- This allows us to keep the history of the project

```
.
├── AUTHORS.md
├── LICENSE
├── README.md
├── poetry.lock                       <- Poetry python package requirements file
├── pyproject.toml                    <- Poetry project config
├── data
│   ├── generated_attention_check_questions <- Temporaray storage for generated attention check ques
│   ├── generated_audio_files_link          <- Temporaray storage for generated audios
│   ├── generated_mcq_tf_questions          <- Temporaray storage for generated mcq/tf questions
│   ├── generated_nlp_stats                 <- Temporaray storage for generated NLP stats
│   ├── generated_speech_score              <- Temporaray storage for generated speech scores
│   └── scraped_data                        <- Temporaray storage for Scraped data
└── src
    └── audioproject                        <- Main project directory
        ├── nlp_utils                       <- All NLP related code
        │   ├── data                        <- Containing test and google vocab data required for go
        │   ├── main_scripts                <- All the main execution scripts
        │   │   ├── questions               <- Scripts for question generation
        │   │   ├── speech                  <- Scripts for Audio and Speech Score generation
        │   │   └── stats                   <- Scripts for NLP stats generation
        │   ├── test_notebooks              <- Some sample notebooks for adhoc usecases
        │   └── utils                       <- Utility code
        │       ├── chatgpt                 <- OpenAI related utility functions
        │       ├── s3                      <- AWS S3 related utility functions
        │       └── stats                   <- Main NLp Stats generation class
        └── scraper                         <- All scraping related code
            ├── cleaning_scripts            <- Cleaning scripts for the scraped corpora
            ├── scraping_scripts            <- All the scraping scripts
            └── utils                       <- Utilty functions for scraping
├── Dockerfile                        <- Docker script in charge of container creation
```

Package Version Management

- Most of the code is written in python and we use different python libraries
- Python libraries are updated frequently and if we try to run the same code again after some time, it might be possible that code might run into failures
- To make the code reproducible, we should be able to run the code with same version of python libraries with which it was developed and tested
- To manage the package versions we use python library called Poetry
- Poetry provides easy way to manage, build and publish python packages. Also, it can create virtual environments to execute the code

Containerization

- Even though we made the code reproducible by using the same version of python libraries, there's still possibility that because of the different execution environment (OS and OS Architecture) the scripts can fail
- To improve the reproducibility of our code, we used docker
- We installed all the required libraries and copied all the required code to the docker
- Anyone can easily build and run the docker container to execute the python scripts
- E.g. `docker run -v ${PWD}/data:/home/audio_user/audioproject/data/ -it audioproject:0.0.1 poetry run python src/audioproject/nlp_utils/main_scripts/stats/generate_stats.py src/audioproject/nlp_utils/data/test.csv id text data/generated_nlp_stats/test_stats.csv`

Documentation

- Even though everything is maintained in the python scripts and docker, we still have to lay out the instructions on how to execute a python script using docker
- We maintained all the required documentation in the github repo readme files
- We described how to get started and explained all the script's execution method and all the required arguments
- E.g

How to generate nlp stats

All the nlp stats generation scripts are under src/audioproject/nlp_utils/main_scripts/stats.

1. src/audioproject/nlp_utils/main_scripts/stats/generate_stats.py this script's input is a csv file containing a unique id and text column containing the text for which you want to generate the text. You need to mention the output file path as well.

- First argument -- input file path e.g. test_file.csv contains all the data
- Second argument -- unique id column e.g. id is the unique column in the file
- Third argument -- column name containing the text data e.g. text is the column which contains the text data
- Fourth argument -- output file path e.g. output_file.csv

```
docker run -v ${PWD}/data:/home/audio_user/audioproject/data/  -it audioproject:0.0.1 poetry ru
```

Data Management

- Once the docker runs the python scripts with the input files, it generates the output in the temporary data directory
- We maintain all our data in the shared google drive or the cloud storage like AWS S3
- All of the temporary data is moved to google drive

Acknowledgment

- Research reported in this presentation was supported by the National Library of Medicine of the National Institutes of Health under Award Number R01LM011975.
- The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.
- Thanks to R4R program which helped my academic research skills and motivated me to work as per the open science approach.
    - Also, meeting with other researchers from diverse field helped me to understand Data Science/ML is being used in other domains
- R4R program is led by Arizona Institute for Resilience (AIR), CyVerse, and the Data Science Institute (DSI).

Thank you!