## Information Retrieval System

### Design choices

One of the requirements for the project was to assess the performance of summarizers on CNN or Dailymail articles. Ultimately we concentrated on the Dailymail articles for this project because the Rouge evaluation needs a human written summary as the reference text, and Dailymail articles included bullet points summarizing the keypoints of an article. Using these bullet points meant that we could fully automate the evaluation process. For CNN articles for which we would’ve had to write our own reference text.

### How it’s built

The information retrieval system (IR) for this project was made using Python packages ”beautifulsoup4” and ”requests”. The module defines the methods needed for retrieving and parsing the information for summarization and evaluation. An URL of an article or path to a html file can be given as input and the method creates beautifulsoup object out of it. A method was created for parsing the the article and finding the bullet points, which the method returns as a list. Another method was created for parsing the article content itself and returning the text as a string object. Finding the bullet points was fairly straight forward, however finding the article contents was trickier because the text was scattered between multiple paragraphs.

## Graphical User Interface walk-through

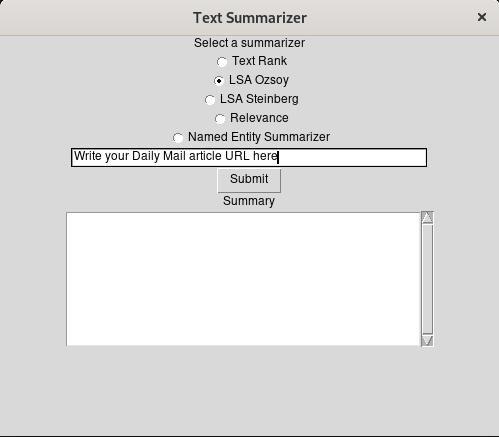


Figure 1: Graphical User Interface input

The graphical user interface (GUI) takes three inputs from the user. First input is radio button between the five different summarizers. The radio buttons are below ”Select a summarizer” help text, and user can select only one of the summarizers for each evaluation.

The second input is the Uniform Resource Locator (URL) field, which defaults to help text ”Write your Daily Mail article URL here”. The URL must not include quotation marks or any other special characters at the start of the text or at the end, and the URL must be a Dailymail URL. There is no input sanitization implemented for this version of the program, so the user must take care writing the URL.

The last user input is clicking the ”Submit” button. This button launches the program calculating the rouge scoring and generating the summary. Under the button there is a text ”Summary” and under it scrollable text field in which the generated summary will appear.

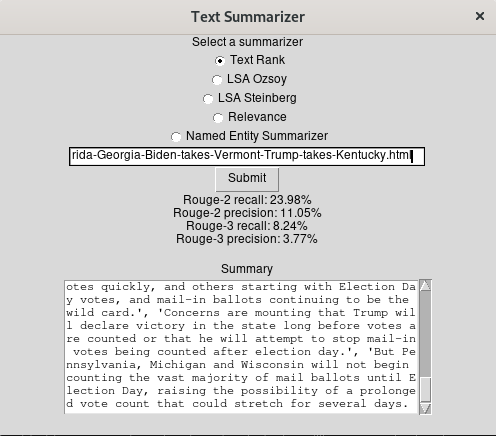


Figure 2: Graphical User Interface output

After the user has pressed ”Submit” button, the program fetches the article using its IR system, creates a summary of the article contents using selected summarizer and calculates the Rouge values using the bullet points found in the article. Pressing the button then displays the output in two fields under the button. First output is the Rouge recall and precision values in percentages for both Rouge-2 and Rouge-3. This field is only revealed after calculating the values. The generated summary will be inserted in the previously blank field. The Rouge values field and the summary field both clean up themselves upon pressing the button again and the user can try out the same article with all of the summarizer.

### Future development

Multi-linguality or even cross-linguality are great features, but left out of the scope for lack of resource, and the application will be langue-specific. This version of the program uses only english language but other languages are supported by the packages used with some tuning. Future version could easily have drop down list from which the user can choose the language that an article uses. An option to automatically detect the language could be used as well, and user could select which language they want the output in.

This version of the program focuses heavily on Dailymail, so in a future version the IR system would detect which site is being processed, and decide which methods to use for parsing.

## Program performance metrics

All the speed performance metrics were gathered using cProfile module and all the memory usage metrics were gathered using memory profiler module.

### Information retrieval

Speed and performance evaluation for the IR were tested using one large article (URL: <https://www.dailymail.co.uk/news/article-8911489/Polls-close-battleground-Florida-Georgia-Biden-takes-Vermont-Trump-takes-Kentucky.html>). The article was retrieved five times using different summarizers selections for evaluating the speed.

On average, the IR system takes 1,8 seconds to perform all the operations, which means opening the article, parsing the article content and finding the bullet points. Most time is spent loading the article from the site, which takes 1,4 seconds on average. Below a sample of the speed metrics, where the columns from left to right are number of calls, total rime, time per a call, cumulative time, time per a call and filename.

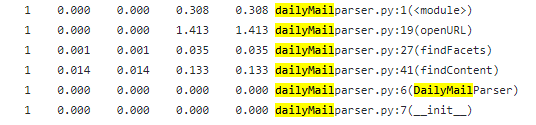


Figure 2: IR system speed

Memory usage is dependant on the article, and for this article the html site was 3,8 megabytes and the beautifulsoup object was 2,4 megabytes, and in total this uses 6,4 megabytes worth of memory. Article content parsing and finding the bullet points don’t increment or decrement the value.

### Summarizers

Speed

Memory usage

### Rouge evaluation

Speed

Memory usage

### Comparing against other applications

Speed comparison was made against three different existing article summarizers, that are hosted on web. The web applications are Resoomer (<https://resoomer.com/en/>), Esummarizer (<http://esummarizer.com/main/summarize>), and SMMRY (<https://smmry.com/>)

### Conclusion on performances