NLP Project: English text-to-code for web development

Guerin Clement

s5355484@studenti.unige.it

guerin.clementpro@gmail.com

1 - Introduction

This project is inspired by OpenAI Codex. This tool is not available for public testing, so I decided to make a simpler version of it. This work is made for my Natural Language Processing (NLP) course at the University of Genova.

This tool is a chatbot you can interact with in a Terminal shell that tries to understand a web page you describe to it and generate the corresponding html code. It uses Bootstrap for the CSS and JS part.

You can download the source code of it from github:

https://github.com/Hvedrug/NLP-project-chatbot_writing_web_page.git

2 - Architecture and requirements

2 – 1 - Project architecture

The project is made of six python source code files, one “read me” text file and a “test.html” that is used to save and visualise the results of the execution of the project. The six python source code files are the following:

Main.py: used to import all the other files, initiate the variables and launch the main function.

Data.py: The file that create all the necessary variables that can be used by the various functions of the code.

htmlObject.py: Contains all the functions necessary to store an html object for the project. By html object I mean a paragraph, a link, a title, an image, a div, …

htmlGenerate.py: Contains the functions to create html code from the stored html objects.

Research.py: All the functions to find an html object in the list of already existing html objects. There is also a function to compare two html objects

userInteraction.py: All the functions to get the user request and extract information from them.

2 – 2 - Requirements and execution

This tool is written in Python (using version 3.7.9 or later) with no additional library.

To execute this tool, open a new Terminal console in the main folder containing the file main.py. Run the following command line:

```
python main.py
```

3 – How does it work

3 – 1 – basic functionality

3 – 1 – 1 – the chatbot

The program prints some text and wait for the user to answer something. When the user gives an answer, the program makes a research by keyword in it and do some actions according to the keyword present in the text. If the user only uses the keyword, in any given order, the action from the program will be the same.

3 – 1 – 2 – Viewing results and closing program

As reminded at the launch of the application, the user can use ‘save’ to generate the html code and save it in the ‘test.html’ file. The user can also use ‘stop’ to terminate the execution. By closing the application, all the unsaved information will be lost.

3 – 1 – 3 – Metadata

By using the keyword ‘metadata’ the user is given the ability to add information to the header of the html document. This metadata can be a title, a description, an author name or a google font used for the all document. The name of the font needs to be the exact name from the google font website (for example: Sofia, Roboto Mono, …)

3 – 1 – 4 – creating an object

The keywords “create”, “make” and “add” let the user create a new object. This keyword need to be associated with the type of the object. In the same message from the user the program will search one of the following identifier: text, paragraph, image, title, link, div, column, row.

The user can also specify in the same sentence:

a colour (for the text if it is text or for the background) by its name (red, blue, yellow, black, white, green, grey, cyan),

a background colour by using background-*colorName* (replace colorName),

the opacity of the object colour with opacity-*Pourcentage* (opacity-10, it needs to be a multiple of 10). The opacity is not currently working,

a size for a title by using a word like (big, small, tiny, medium, average, primary, secondary, tertiary),

one or many Bootstrap class with class:*oneClassName*,

one ID for the object with id:*oneID (not recommended)*.

3 – 1 – 5 – modifying objects

By using one of the keywords “find”, “modify”, “change” the user will be able to modify an already existing element. By default, it is the last one. The user can specify the type of the object for example “last paragraph”. The program is supposed to recognise “the last” and “all” but the later one is not implemented. By default, if the user precise the type of the object it will take the last object with this type.

It is also possible to specify the ID of the object and some class. Class will be added to the object list of class after being compared to the already existing ones to avoid having two class with effect on the same property (colour of the text for example).

3 – 1 – 6 – deleting objects

By using the keyword “delete” the user can delete the last element created, by adding the type of object it can delete the last object of this type (paragraph, text, title, …) unless it is a div object (it would create too many complications with object placement). By adding “all”, the user can delete all the elements created before.

3 – 2 – More complex functions

3 – 2 – 1 – Object placement

To choose the place of an object the program uses the Bootstrap grid and the CSS class that goes with it. These class are named “col” and “row”. Three ways have been implemented to handle this.

The first way is the most precise one, but it is not user friendly. Th user can create “div” object and attribute them the correct Bootstrap class. The user needs to also close all div object himself (with the command “close div”), if not, all the object will be inside the

previous one. The Bootstrap classes are “row-*number*” and “col-*number*” where *number* is an integer between 1 and 12.

The user can also use the keywords row, col and column. The user is supposed to always create the columns inside a row (it is the way Bootstrap is working). If the user creates two column they will be placed next to each other. To get nested columns the user should add the keyword “inside”. It is working the same way with rows, adding a second row will close the open column and row and open a new one. Note that add more than 12 columns this way will have same visual effect has adding a new row because Bootstrap grid length is 12 units.

Finally, the user can create rows like before and specify a size for columns with one of the following keywords: big, small, half, tiny, quarter, third, three-quarter, two-third, … These can also be created automatically by the program for objects such as paragraph, title and images (not the other ones). To do so, the user can use sentences like “add a title in the middle”. Note that this only let the user place one object per row. The usable keywords are left, right, middle, far-left, far-right.

4 – Examples

In this part we will see some sentences the user can write and their results.

A sentence can trigger some actions from the program, this one can request more information to the user (the content of the title for example). In this part I’ll use “>” to say that the following content needs to be given to the chatbot after the current one. I’ll also use “” to say press the enter key to skip this step.

Note that the chatbot will give explanations on what it’s waiting for if needed, and it will also say what it has understood from previous sentences and what it’s doing with it.

4 – 1 – creating a simple title and paragraph

create title > 1 > NLP Report
add paragraph > This is my NLP Project report
modify the last text to be blue > “”
save

4 – 2 – add centered image using keywords

add a new row
add a new column of a third of the page
add a new column of a third of the page
add an image > test.jpg > “”
save
(note that test.jpg is an image already existing in the same folder as main.py)

4 – 3 – add more complex example

add a big title in the middle > NLP Report
create a small title on the left > this is a small title
change the title to blue
add a text > Hello
add an image on the right > test.jpg > “”
delete the last paragraph
metadata > font > Sofia
save

5 – Limitations and improvements

5 – 1 – Limitations

There are only five colors usable because of the way Bootstrap handling them. It could have been interesting to implement more but 5 were enough to make basic tests. The opacity could help to bring more nuances, but it isn't working now.

When using complex placement, some modification of objects doesn't work anymore. For example, if the user "add a title in the middle of the page" and then want to "change its color to blue", because the title is in a row, the program will try to change the color of the div. The user needs to use "change the last title color to blue".

There are some sentences that are not recognized but have unnatural equivalents. For example, if we are speaking of a title "make it bigger" does not work but "change the last title to be bigger" should let the user modify it. "make" is a keyword to create not to modify existing objects.

The sentences to modify the last element don't always work because some keywords are searched only for one type of object. By not specifying the type in the sentence the keywords will be ignored.

5 – 2 – Possible improvements

It could be interesting to find a way to understand some context between words. For example, being able to use "make" to create and modify depending on the context. Using "make" without an object type to modify the last object.

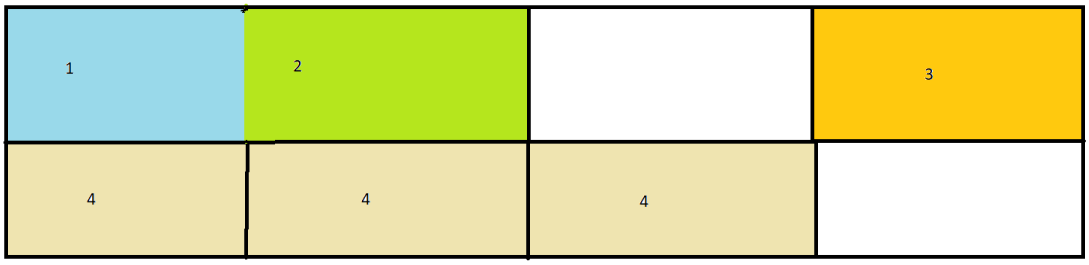
Implementing more html objects to create more useful web pages: menu bar, buttons, tooltips, … The hard part is that some of them have nested objects inside them. The goal of the project was not to implement an interpreter for the all Bootstrap framework, so I focused more on the understanding part.

Implementing a function that would compare the words with a list of correct words to give a similarity score (like the Levenshtein distance) to correct some miss spelled words and improve the efficiency of the program.

Implementing a way to move objects after they have been placed. To do so it would be necessary to entirely change the way placement is handled. One solution could be to represent the page as a grid. Instead of placing objects inside rows and columns, it could be interesting to give them coordinates and store in an array the ID of the elements that

should be in the cells of the grid. Then making a loop over the array and creating the rows and columns as required.

Example: [[1,2], [], [3]],
[[4], [4], [4], []]



6 – Conclusion

This project main goal was to create a chatbot able to understand the description of a web page and return the html code of this same page. The program is able to understand request such as creating, modifying, deleting and placing objects on a web page. There is also the possibility to handle sizes and colors. However, there is only a limited number of objects available (texts, pictures, links), complex objects are not available. Modifying placement is not possible either. The web pages generated are not very complex but there is a still some level of understanding almost only focused on keywords and a little of context.