Word Cloud Webserver

Wyatt Greenman and Sarah Fondots

Introduction

Our project's goal was to create a webserver that hosted a word cloud generator. The webserver has to have at least two nodes, one to run the word cloud generator function, and one to display the word cloud after it has been generated. The word cloud generated would use pre-generated user selected text files to generate the final image. Additionally, there would be the option for the user to enter their own text, which would be added to a separate file and be used to generate a unique cloud. The webserver would be used to launch a Jupyter notebook through docker where all the files needed to make this work would be run from.

Webserver

If a user wants to launch the webserver used in this project they need to follow these steps. To start with, the user should find a service that they want to use to launch the server. If the service does not already have docker installed, then the user should use the install_docker.sh file from the github repository to install docker. After this the user should ensure they know the ip address for their instance on their service of choice and then run the command docker-compose up to launch the image stored in the docker-compose.yml from the github repository. This docker-compose.yml launches an image of Jupyter/Spark and provides the user with a link to follow to access the Jupyter notebook. The user will want to use the last link and replace the local ip address (127.0.0.1) with the ip address for the user's instance on the service they are using. This will allow the user to load up the files for the word cloud generator into a fresh instance of Jupyter notebook or it can be used for anything else you might want to use Spark and Jupyter for.

Docker is the primary tool used to launch the webserver, which makes getting the webserver up and running very easy. It took a little work and a lot of research to edit the docker-compose.yml file to get it working the way we wanted it to work. Once the docker-compose.yml file was working properly, it only took a few commands to get the Jupyter/Spark environment to launch and that means it can easily be deployed by any service that is capable of installing and running Docker. This freedom to launch the service from nearly anywhere is a major benefit to the project as a whole.

Word Cloud

To create the word cloud we began testing in a local python IDE and then moved our code into a Jupyter notebook after development. Using the wordcloud, matplotlib and pillow packages we were able to create the word clouds themselves and the GUI was created using tkinter. Since wordcloud, matplotlib and pillow packages are all non-native python packages they needed to be added to the python build we deployed on our Jupyter notebook, this can be done by running the following code:

```
!/opt/anaconda3/bin//python -m pip install wordcloud
!/opt/anaconda3/bin//python -m pip install numpy
!/opt/anaconda3/bin//python -m pip install pillow
!/opt/anaconda3/bin//python -m pip install matplotlib
```

This only needs to be run once and can then be commented out. After these packages are installed it is time to build the word clouds, since we are using both preloaded as well as user

generated text we must first download the test we are interested in as txt files, these can then be saved in the projects Data file to be read later. We also chose to use an image overlay for our pre-loaded text we will download those as well, in order to work with the code that we have written it is best if these images are basic black and white jpeg, jpg or png files.
Once we have our files together it is a relatively simple process to construct our word cloud. First we have python read in our file and assign it to a variable we do the same with the image we plan to lay over our finished word cloud using Martin Luther Kings I have a dream speech the code should look something like this:

```
    dream = open("DreamSpeech.txt”).read()
    dream_mask =np.array(Image.open("comment.png"))
```

Once the files have been read it we use the wordcloud package to bring everything together. This package allows us to set a background color as well as other graphical options. We will be keeping our word simple with a black background and using our image as the mask overlay the code used to create the word cloud should look something like this:

```
    dreamcloud = WordCloud(background_color='black',
            contour_width= 1,
            contour_color='black',
            stopwords=stopwords,
            mask=dream_mask).generate(dream)
```

The stopwords variable is an option that causes the wordcloud package to ignore common words used to construct sentences like and, or, the as well as others. Finally, to output our word cloud we run the following lines of code:

```
    plt.imshow(dreamcloud, interpolation='bilinear')
    plt.axis("off")
    plt.tight_layout(pad=0)
    plt.show()
```

This will generate our word cloud beneath the code chunk in our Jupyter notebook, to create the rest of our word clouds we simply repeat the process using the other files and images.

While using the Jupyter notebook to create static word clouds can be interesting and is certainly a useful tool when visually representing text data, it is by its nature a bit immutable so I also wanted to create a way for a user to generate a word cloud using their own text. To do this I created a simple GUI using the tkinter package that is native to python, there are other more aesthetic GUI packages available but I unfortunately have a limited amount of experience with them. We begin our GUI in much the same way we did our static word clouds by letting python know where to find our data so that our prebuilt word clouds can still be rendered. After we have finished with the basic structures we have used before we augment them to take user commands by adding buttons and a text field. This is done by creating a function we will call button_enter and should look something like this:

```
    def button_enter():
        text = e.get()
        wc = WordCloud(background_color='black',
                contour_width=1,
                contour_color='black',
                stopwords=stopwords)
        wc.generate(text)
```

```
plt.imshow(wc, interpolation='bilinear')
 plt.axis("off")
plt.tight_layout(pad=0)
plt.show()
```

Where the command text = e.get() retrieves any text that has been entered into our text field. I had hoped to be able to run this through Jupyter like our original basic word cloud file however I ran into issues getting the GUI to output and so far I have only been able to run the file from the command line.

Have the goals been met?
We met many of the goals that we originally set out for ourselves, at the conclusion of our project we did create a webserver that would run both our Jupyter notebook file as well as our GUI. I was disappointed we were not able to construct a database to go along with our project to expand on the files that we had available to generate word clouds and to store user generated text between sessions. I intend to continue modifying this project in the future with this goal in mind however at the time of writing these features are unavailable.

Conclusion
This project was an opportunity to use skills that we have learned in this class as well as others to create a final product that spanned a full course of development from planning to implementation. Even losing significant time and the ability to meet and develop in person due to the outbreak of Covid-19 was a learning opportunity, many projects are developed by diverse staffs often spread out around the world and learning to work under those sorts of conditions is invaluable. We have created a final product that I look forward to improving and augmenting in the future for other data analytics tasks improving not only the functionality but also the scale that our implementation can manage.