Practical exercises II

Due on: Tuesday, January 11, 2022

This assignment consists of two tasks, using python's Flask library to make our NLP script available in a web service and another front end exercise to become more familiar with *Vue.js*

1. Flask exercise

In this exercise you will use Flask to build a simple web server that calls spacy NLP functions (like in your first script) to process text input received from an HTTP request.

To get started, download the project template from moodle. You will need to set up a python (virtual) environment in which to install Flask, spacy and the *en_core_web_sm*-pipeline that you've used before. Let's examine the project structure:

- *app.py*: Your basic Flask app that you will need to run
- /nlp
  - *utils.py*: Contains all NLP-related functions using spacy
- /templates
  - *about.html*: An info page for your app
  - *illegal.html*: A notice page for "illegal" requests

Your task is now to complete the two python files using the following instructions and the comments inside each file:

1. *app.py*: For each of the functions add a meaningful route decorator and fill out the function body to achieve the intended behaviour.
2. *utils.py*: Complete the two function templates, you can add helper functions and variables if necessary.

Manually test your Flask app by sending valid and invalid requests to it, using your browser and an http request tool of your choice (curl, browser plugins, standalone apps...)

[Optional, open ended]: Sending the text as plain text/html data for processing can be a bit tedious, for larger texts it would be better to be able to send text files to your app. Processing large texts with spacy can take a while however. Explore and try to implement any number of ways of making your app more practical:

- Extend your app to accept and process text files.
- Limit the maximum text size for requests.
- Have your app store a few processing results so that you don't have to use spacy to process repeated request for the same text (consider hashing your input texts for this)
- Check the Tips for efficient processing in spacy and implement them.
- Anything else you can think of.

Please submit your code either as a zip archive on moodle or just provide a link to a github repo.

2. Vue.js exercise

In this exercise you will explore different ways of highlighting text to the user. Introduction videos and template code will be made available on moodle.

If you inspect the html file you can see a big textarea that can be used to input text for highlighting and a small text input to specify which string to highlight in the input text and a color input for picking a highlighting color. Below them you should see three div containers for displaying highlighted text.

1. The first container should highlight text using `<mark>` tags
2. The second container should highlight text using `<span class="highlight">` tags instead, their style can be edited in the *style.css* file.
3. **[Optional]:** Executing unsanitized user input as html is not a great idea (look up XSS), try to find an input for the textarea that causes your app to display an alert.
4. For the last container, find a different highlighting approach using the v-for directive to create a sequence of spans with conditional highlighting in the color that is selcted in the color input element. To do so, you can split the input text into an array of objects that contain the text chunks and an indicator for each object whether its text chunk should be highlighted. It may be helpful to start by manually creating an array in your app data to test your highlighting functionality and then add a function that transforms the text input to the required format later.

[Optional]: Refactor the solution for the previous exercise using custom Vue components. You will probably want to have a component for adding tasks and a component for list items but you can choose any component architecture you like.

As with the NLP exercise, please submit your code either as a zip archive on moodle or just provide a link to a github repo.