citibeats

SOFTWARE ENGINEER

HOME ASSIGNMENT

Thanks for applying for this position!

The goal of this assignment is to evaluate your performance on day-to-day tasks related to the position you're applying to.

Disclaimer: Citibeats will NOT use your answers to solve any current work problem or need. This is NOT a way to get you doing free work.

Instructions to complete the assignment:

- Take the time to complete these tasks properly. There's no set deadline.
- Feel free to use the Internet to complete your task, the same way you would on a workday.
- When you're done, please send your code in a ZIP file via email to the person that sent you this assignment.

Evaluation process:

- You'll find the maximum points for each task in this document.
- We're more interested in your reasoning, rather than correct solutions.
- Tasks can score negative points, so providing wrong solutions affects the score.

Good luck!

Tokenizer challenge

Context: in most NLP applications, one of the first steps is splitting the text into words. This process is commonly known as tokenizing.

We want to build a tokenizer application that, given a text and its language, returns an array of words.

Example:

```
INPUT:
Agent Smith: As you can see, we've had our eye on you for some
time now, Mr. Anderson.

OUTPUT:
Agent
Smith
As
you
can
see
we
have
had
our
eye
on
you
for
some
time
now
Mr.
Anderson
```

Notice that most punctuation has been cleaned, except the one with meaning, like "Mr." We have also expanded the contraction "we've" to "we have".

Task 1 - Multilingual tokenizer [2 points]

Implement a program that given a text and its language, returns a list of words.

It should work at least in English, French, Portuguese, and Spanish.

The code must be written in Python. You can use any libraries such as spaCy, NLTK, or others. It should also include some unit tests.

NOTE: feel free to make any decisions regarding how a text should be tokenized. If you think it's necessary, provide an explanation of your reasoning, we'd love to hear it!

Task 2 - API [2 points]

We want to be able to deploy the tokenizer as a micro-service that can be invoked via HTTP.

Encapsulate the program of Task 1 into an API that receives two inputs and returns a list of words in JSON format.

Example:

```
INPUT:
{
    "text": "Agent Smith: As you can see, we've had our eye on
you for some time now, Mr. Anderson.",
    "lang": "en"
}

OUTPUT:
{
    "tokens": [
        "Agent",
        "Smith",
        "As",
        "you",
        "can",
        "see",
        "we",
        "have",
        "had",
        "our",
        "eye",
        "on",
        "you",
        "for",
        "some",
        "time",
        "now",
        "Mr.",
        "Anderson"
    ]
}
```

Define the endpoint(s) of the API as you prefer. Remember to validate the inputs and return an appropriate error response.

The code must be written in Python. You can use any frameworks such as Flask, Django or others. It should also include some unit tests.

Please include a short reference of how to invoke the API.

Task 3 - Dockerize & deploy [1 point]

Dockerize the application and add a README.md with instructions to build and run the image.

Deploy the application in a hosting environment that can be called publicly. We recommend using Heroku which has a generous free tier. Feel free to use any other hosting solution if you prefer.

Bonus Task - Detect language [1 point]

This is an extra task to obtain one additional point.

Make the language parameter optional. The program must be able to detect the language and return it.

Example:

```
INPUT:
{
    "text": "Agent Smith: As you can see, we've had our eye on
you for some time now, Mr. Anderson."
}

OUTPUT:
{
    "lang": "en",
    "tokens": [
        "Agent",
        "Smith",
        "As",
        "you",
        "can",
        "see",
        "we",
        "have",
        "had",
        "our",
        "eye",
        "on",
        "you",
        "for",
        "some",
        "time",
        "now",
        "Mr.",
        "Anderson"
    ]
}
```

Note that users must be able to specify the language if they choose too. In that case, the language detection should be bypassed.