Teste Prático - Inteligência Artificial - Nuveo

Este é o teste prático para candidatos à vagas no time de Inteligência Artificial da Nuveo.

O objetivo do teste é fornecer aos candidatos uma oportunidade de demonstrar conhecimentos na área de visão computacional e desenvolvimento em Python.

Instruções Gerais

- Deve ser apresentada uma solução para cada problema proposto
- Crie um repositório Github, com o nome `nuveo-teste-ia` e desenvolva suas soluções.
- Submeta o link para o repositório por e-mail, em resposta ao e-mail recebido com o link para o teste.
- Caso sejam necessários arquivos auxiliares muito grandes, estes podem ser compartilhados em nuvem (via Google Drive ou Dropbox, por exemplo), desde que haja instruções sobre qual o caminho relativo para tais arquivos na solução final.

Estamos cientes que o tempo costuma ser um fator limitante. Porém, a avaliação levará em consideração todos os aspectos das soluções apresentadas, como criatividade, estratégia de raciocínio, documentação de código, estrutura de código, além do resultado obtido. Portanto, faça o possível para que as soluções apresentadas reflitam ao máximo o seu conhecimento!

Problemas

01-WheresWally

Para este teste, será criada uma solução capaz de identificar um objeto de interesse em um conjunto de dados.

Descrição

No fundo do oceano? No topo da mais alta montanha? Onde o camarada Wally pode estar? Há uma forma de detectar Onde Está Wally automaticamente?

Um conjunto de imagens com fundos aleatórios é fornecido. Em cada imagem, a mesma foto de Wally foi adicionada em uma posição e rotação aleatórios, com uma leve distorção de perspectiva.
Os seguintes itens foram fornecidos como conjunto de treino (`TrainSet`):

- Uma coleção de imagens aleatórias com a foto de Wally em uma posição aleatória
- Um arquivo json com anotações (padrão LabelMe), determinando a posição da foto de Wally em cada uma das imagens de treino.

Os seguintes itens foram fornecidos como conjunto de teste (`TestSet`):

- Uma coleção de imagens aleatórias com a foto de Wally em uma posição aleatória

Os seguintes itens foram fornecidos como conjunto de referência (`ReferenceData`):

- A foto original de Wally
- Um arquivo CSV contendo o centróide da foto de Wally em cada uma das imagens de treino

Objetivo

O teste tem por objetivo avaliar o conhecimento em visão computacional e inteligência artificial, em especial o uso de ferramentas para criação de uma solução que seja capaz de identificar um elemento nas imagens de um conjunto de dados. Deve ser desenvolvido um método para encontrar Wally automaticamente em cada imagem em `TestSet` . O resultado final deverá ser um arquivo CSV contendo o centróide em cada uma das imagens de teste, de acordo com as especificações.

Dados

Dados em arquivo zip hospedado no Google Drive. Acesse aqui

Instruções

- Como comum em cenários reais, o dataset pode conter erros. Estes são intencionais e os candidatos e candidatas são livres para tratar os mesmos da melhor forma que preferir.
- O dataset foi dividido para que dados desconhecidos sejam utilizados nos testes. Foram separados 20% dos dados de forma aleatória para preparo do conjunto de teste.
- As anotações estão no padrão `LabelMe`. Pode encontrar o software aqui
- O arquivo CSV tem o nome do arquivo da imagem na primeira coluna, a posição em `x` do centróide na segunda coluna e a posição em `y` do centróide na terceira coluna
- Foram fornecidos recursos diferentes para que abordagens diferentes sejam possíveis. Dependendo da abordagem escolhida, nem todos os dados fornecidos serão necessários.
- Replique o padrão do CSV de referência na submissão. Isto é importante para a avaliação.

Requisitos

- O teste não exige um método de processamento de imagem pré-definido. O candidato ou candidata é livre para escolher qualquer tipo de fluxo de processamento de imagens para alcançar o melhor resultado.
- O código fornecido deve atender aos seguintes requisitos:
  - Deve ser documentado, facilitando entendimento pelo avaliador.
  - Caso seja utilizada técnica de IA na solução, código utilizado para treino de modelo deve estar presente na submissão.
  - Deve conter instruções de como preparar o ambiente de execução e como executar o código.
  - Deve conter uma descrição da solução proposta, com descrição dos métodos, premissas, limitações e quaisquer outras informações relevantes (arquivo `markdown` é recomendado).
  - Não é necessária uma estrutura pré-determinada nesta solução. Solução pode ser apresentada como projeto Python, entry point em Docker, Jupyter Notebook, etc.
- Caso seja necessário poder computacional para treino de modelo, ferramenta Colab pode ser boa alternativa. Caso algum código seja desenvolvido no Colab, deverá ser identificado e apresentado na solução final como Jupyter Notebook.
- As soluções devem ser fornecidas conforme instruções, pois é importante para os scripts utilizados durante a avaliação de resultados.

02-SMSSpamDetection

Para este teste, será criado um projeto Python utilizando um modelo pré-treinado.

Descrição

O dataset utilizado na composição deste teste tem por objetivo auxiliar estudos de detecção de mensagens SMS de spam. O dataset contém 5574 mensagens SMS em inglês no total. Cada mensagem foi classificada em `ham` (mensagem legítima) ou `spam`.

São fornecidos dataset de treino, validação e o dataset de teste. Tratam-se de arquivos CSV em que cada linha corresponde a uma mensagem SMS completa, sem identificação de classe.

Exemplos em dataset de treino e validação:

```
ham    Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amo
spam    SMSSERVICES. for yourinclusive text credits, pls goto www.comuk.net login= 3qxj9 unsubscribe with STO
spam    25p 4 alfie Moon's Children in need song on ur mob. Tell ur m8s. Txt Tone charity to 8007 for Nokias
spam    U have a secret admirer. REVEAL who thinks U R So special. Call 09065174042. To opt out Reply REVEAL
```

Exemplos em dataset de teste:

```
Dunno y u ask me.
K.k:)advance happy pongal.
I know but you need to get hotel now. I just got my invitation but i had to apologise. Cali is to sweet for m
Do you know what Mallika Sherawat did yesterday? Find out now @  &lt;URL&gt;
```

Os datasets de treino e validação foram utilizados para treinar um modelo de detecção de spam, que recebe uma string como entrada e retorna uma probabilidade de que a mensagem de entrada seja spam. Uma instância deste modelo foi salvo como `"Model/sms_model_v1.pkl"`, podendo ser carregado como objeto Python utilizando

```
import pickle

sms_model = pickle.load(open("/path/to/Model/sms_model_v1.pkl", "rb"))
```

O modelo foi treinado em Python 3.6.12, utilizando
`sklearn==0.24.1`

mas deve ter compatibilidade com outras versões próximas.

Objetivo

Para este teste, o candidato deve servir um modelo pré-treinado, que será consumido por um cliente na avaliação. O candidato deve focar na estrutura de código e boas práticas de desenvolvimento, enquanto atende às especificações da solução.

Dados

Dados em arquivo zip hospedado no Google Drive. Acesse aqui

Instruções

- A solução entregue deve ser um projeto Python. Este projeto deve conter uma classe `SpamDetector`, com dois métodos públicos que recebem uma `string` contendo uma mensagem de SMS como entrada.
  - Um método `prob_spam`, que deve retornar a probabilidade, entre 0 e 1, de que a mensagem de entrada seja `spam`.
  - Um método `is_spam`, que deve retornar um valor booleano de `True` se a menssagem for um `spam` e `False` se a mensagem for um `ham`.
- Na avaliação, a inferência para cada mensagem do dataset de teste será feita em um script, a partir da criação de uma instância da classe `SpamDetector` e dos métodos `is_spam` e `prob_spam` criados. Por isso é importante que os métodos estejam expostos com esta estrutura.
- A solução deve conter um módulo com testes unitários.

Requisitos

- O teste não exige um método de processamento de imagem pré-definido. O candidato é livre para escolher qualquer tipo de fluxo de processamento de imagens para alcançar o melhor resultado.
- O código fornecido deve atender aos seguintes requisitos:
  - Deve ser bem documentado.
  - Deve possuir testes de unidade (`unittest` ou `pytest` são recomendados).
  - Deve conter instruções de como preparar o ambiente de execução e como executar o código.
- As soluções devem ser fornecidas conforme instruções, pois é importante para os scripts utilizados durante a avaliação de resultados.