Website Similarity using Deep CNN

劉仲昀
F74086145

吳佩恩
F74086064

Abstract

A website may look a little different on two webdrivers. For example, some objects are dynamic part like advertisements that change from time to time. If we compare two images by traditional methods, we will occur the new problem that is the text font of Chrome and Firefox are not the same which may lead to the failure of traditional methods.Therefore, we want to using Deep CNN to learn where are the differences of one website on two webdrivers.

1. Introduction

Our approach uses Siamese Network to recognize whether if the two images are similar. The two images are fixed-size and crop from the snapshot of websites. One snapshot is from Chrome, the other is from Firefox. The reason why we choose Chrome and Firefox is that the Chinese text font is different on these two driver.

2. System framework

2.1. Dataset

We will use web crawler to collect our dataset. First, we snapshot two images from Chrome and Firefox. Then, we artificially circle the different of two images. The most of the websites may be similar on two webdrivers. That means we will collect a lot of positive pairs and a small amount of negtive pairs. To solve this problem, we will take two images from different website to generate negtive pairs.

Second, we crop one of the snapshot into several fixed-size pieces (or is called by template). Each template will find out the most similar position in the other snapshot by using "match template" and then crop it down. If any of these two piece is including the circled part in previous step, it will be counted as negative pairs. Otherwise, it will be counted as positive pairs.

In average, one website can gain about 150 pairs. Therefore, we decide to label and circle about 100 websites, which not including the strategy of taking two images from different website to generate more negtive pairs. We estimate that we total will have 10000 each for positive pairs and negetive pairs.

We finished the program that will automatically crawl the images from the website. We have collected about 60 websites, which is far from the target amount, because we decided to use small amount of data to practice training the Siamese Network first. This helps us to realize the problem we may encounter in future.

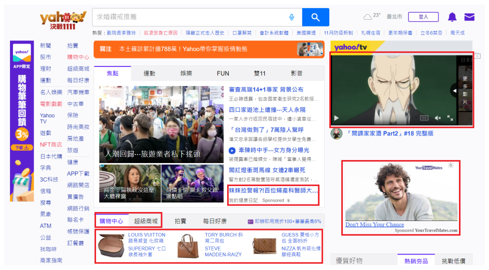

Figure 1. Snapshot for Chrome

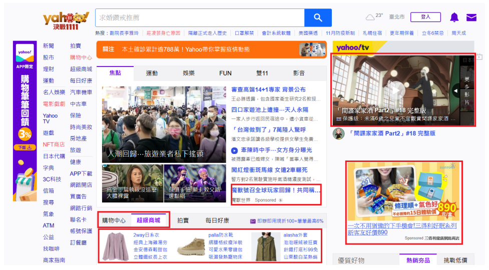

Figure 2. Snapshot for FireFox

2.2. Siamese Network

We used a Siamese network architecture [2] with 2 CNN's whose weights are shared and they are trying to minimize the loss function. A Siamese network is a function f that maps each image I into an embedding position x, given parameters θ .x = f (I; θ).

The parameter vector θ contains all the weights and biases for the convolutional and inner product layers, and the amount of parameters depending on the size of the network. The goal is to solve for the parameter vector θ such that the embedding produced through f has desirable properties and place similar images nearby.

The input to the network is pairs of images. We can map these images through our network to get embedding's x_1, x_2. If the network had learnt a good embedding, we would find that (x_1, x_2) are nearby if these two input image are similar.

Before we collect the enough amount of website data, we have trained a basic Siamese network model using Resnet18 with the MNIST dataset. With the result value we can easily judge that if two images of hand-write number are the same number. The final accuracy is 99%.

After training this model, we know how to produce the training process of the Siamese network with the data and the other settings. Next step, we will try to use different backbones in the network about Resnet18 to VGG16, and also apply the dataset of the website screen shots generated by ourselves.

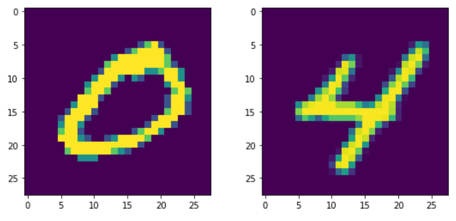

Figure 3. In the test, the result value of these two images are $1.7620e^{-7}$. (If the model's output value is less then 0.5, it means these two images are not the same number.)

2.3. Loss Function

The network loss can be formalized as the contrastive loss function L [3] which measures how close f is able to place similar images nearby and keep dissimilar images further apart. For one training image, the loss is defined as shown in Eq.1. Setting m to some value should not impact

$$L(\theta)=(1-Y)\tfrac{1}{2}\{D(x_1,x_2)\}^2+(Y)\tfrac{1}{2}\{max(0,m-D(x_1,x_2))\}^2(1)$$

Eq. 1 Contrastive loss function (L). computes loss per training example. Total loss is summation over all image pairs. m=1 (value of m does not impact learning as D would simply scale accordingly).

learning, distance metric D would simply scale accordingly. Label of Y=1 is given to dissimilar images or negative pairs and label of Y=0 to similar images or positive pairs. The two CNN's of the Siamese network have shared weights optimized by contrastive loss function L.

3. Expected Result

Given two snapshots as an input, it will circle the different of two snapshots. Since we crop snapshot into pieces in process, the result will look like a several square which is not matching in the other snapshot. The more similar to the red rectangle we drawed, the better the model is.

4. References

1. Image similarity using Deep CNN and Curriculum Learning https://arxiv.org/ftp/arxiv/papers/1709/1709.08761.pdf

2. Siamese network features for image matching https://ieeexplore.ieee.org/abstract/document/7899663

3. Dimensionality Reduction by Learning an Invariant Mapping. https://ieeexplore.ieee.org/document/1640964