

Natural Language Processing

Project 1: OCR Error Correction using Character Based Language Modeling



Project Description

The objective of this project is to apply Language Modelling to improve the output quality of an optical character recognition (OCR) system.

The OCR baseline system

The [IAM Handwriting Database](#) contains forms of handwritten English text which can be used to train and test handwritten text recognizers and to perform writer identification and verification experiments.

A Neural Network (NN) OCR system which maps an image (or matrix) M of size $W \times H$ to a character sequence (c_1, c_2, \dots) with a length between 0 and L , was implemented for this task. The text is recognized on character-level, therefore words or texts not contained in the training data can be recognized too (as long as the individual characters get correctly classified).

$$NN: \underbrace{M}_{W \times H} \rightarrow \underbrace{(c_1, c_2, \dots, c_{1n})}_{0 \leq n \leq L}$$

Equation 1: The NN written as a mathematical function which maps an image M to a character sequence (c_1, c_2, \dots) .

You can download the model from Canvas where you can find the file Project1NLP_Model.zip that contains a script “run.py”, a folder “model” and some test files.

To use the pre-trained model you need first to pre-process the input images to transform them to IAM format where you need to:

- crop the image;
- increase its contrast and
- thicken the lines



Figure 1: Pre-processing steps of an input image.

You can use [ImageMagick](#) (mainly convert) and [Open CV](#) (cv2 in python) for this task. Once your input image is ready, you can use the given model by running the following command: **Python run.py**

You need to make sure that you are updating paths for the working directories “ExpDir” and “modelDir”, and if necessary the files paths as you can see in the following figure.

```
In [3]: ExpDir = '/Users/haithem.afli/Desktop/CIT/Research/IndustryProjects/Unitek/NN_OCR/run_model/'
print ('Experiment Dir is:' + ExpDir)
modelDir = '/Users/haithem.afli/Desktop/CIT/Research/IndustryProjects/Unitek/NN_OCR/run_model/model/'
print ('Model is in:' + modelDir)
#You need to change this path with your local path to the saved model I provides with the code

Experiment Dir is:/Users/haithem.afli/Desktop/CIT/Research/IndustryProjects/Unitek/NN_OCR/run_model/
Model is in:/Users/haithem.afli/Desktop/CIT/Research/IndustryProjects/Unitek/NN_OCR/run_model/model/

In [4]: class FilePaths:
        "filenames and paths to data"
        fnCharList = modelDir+'charList.txt'
        fnAccuracy = modelDir+'accuracy.txt'
        fnInfer = ExpDir+'test.png'
        # You need to change thses Paths with your local Paths to charList.txt,
        #model/accuracy.txt and test.png files I provides with code
```

Figure 2: Paths you need to modify before running the pre-trained model.

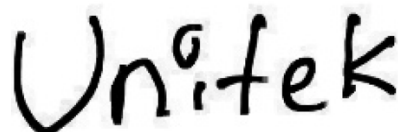
You need, also, to install the following python libraries if they are not already installed in your python3 environment.

Needed Libraries

```
In [2]: import random
import sys
import numpy as np
import cv2
import editdistance
import tensorflow as tf
```

Figure 3: Python libraries you need to install before running run.py script.

Executing run.py script will give the following result if you use the given “Unitek” image.



```
(tensorflow) COM-IT118-32148:run_model haithem.afli$ python run.py
Validation character error rate of saved model: 13.956289%
Python: 3.6.6 |Anaconda, Inc.| (default, Jun 28 2018, 11:07:29)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)]
Tensorflow: 1.11.0
2018-10-14 21:00:52.375282: I tensorflow/core/platform/cpu_feature_guard.cc:141] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2 FMA
Init with stored values from /Users/haithem.afli/Desktop/CIT/Research/IndustryProjects/Unitek/NN_OCR/run_model/model/snapshot-32
Recognized: "Unetek"
```

As you can see the system recognises the word “Unetek” from the input image. It’s a good result but we still can improve it by trying to correct the misrecognised **i** (recognised as **e**).

Your task in this project is to add a new Error Correction module that takes the recognised word and try to correct it using a character/word-based language model. This includes detecting characters given an image containing a line of text and denoising the output of the characters by using a language model.

The Error Correction Module

Your model will use the output of the OCR system and attempts to match each word with a dictionary. You can take the following procedures:

1) Decontraction of the string where you can use [pycontractions](#),

```
I'd -> I would  
I'd -> I had
```

2) Tokenization of the string (word of phrase) where you can use [NLTK](#),

3) for each word:

3.1) Check if the proposed word is an English word, if not, suggest possible words.

You can use the English version of [Europarl](#) corpus for training your Language Model and the Keras (a Deep Learning python Library) implementation of a character based LM explained [here](#).

3.2) Choose the word with the shortest weighted edit distance,

You can use the [Weighted Levenshtein library](#) to calculate the distance between the given word and the suggested words.

4) Contract texts back to its original form. This step simply detokenizes and contracts the words (if it was originally decontracted).

Note:

-You need to add “Unitek” to your lexicon (training data) as it’s not automatically included in the English vocabulary.

-You can run this system on Colab:

<https://colab.research.google.com/notebooks/welcome.ipynb>

References:

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, Andrew Y. Ng. Neural Language Correction with Character-Based Attention.

<https://arxiv.org/pdf/1603.09727.pdf>

Wassim Swalieh, Language Modelling for Handwriting Recognition.

<https://tel.archives-ouvertes.fr/tel-01781268/document>

Ido Kissos and Nachum Dershowitz, OCR Error Correction Using Character Correction and Feature-Based Word Classification.

<https://arxiv.org/pdf/1604.06225.pdf>

Masaaki NAGATA, Japanese OCR Error Correction using Character Shape Similarity and Statistical Language Model.

<http://www.aclweb.org/anthology/C98-2147>

Submission

You are required to submit:

- An electronic copy of all source code developed (.tar.gz or .zip archive).
- Please put **your student name and number at the top of your file(s)**.
- It is your responsibility to make sure you upload the correct file.

All files can be submitted with Canvas before **Sunday Oct 27th 2019**. Please ensure to include **your name and student number** on the python code.

Code

All code should be completed using Python as the programming language.

Your code should have a logical structure and a high level of readability and clarity. Please comment your code and put all code into functions. Your code should be efficient and should avoid duplication.

Late submissions

If you don't get the assignments done to your satisfaction and don't meet the minimum requirements by the deadline, you have the option (as with any assignment at CIT) of submitting up to 1 week late for a penalty of 10%.

This penalty is subtractive. Work that would have earned 55% if on time, would get 45% (not 49.5%) if late.

The penalty is applied weekly. So, 1 day late costs the same 6. If you want to take that option please let me know. Otherwise, I will just correct whatever I have.

If you have a specific reason for submitting a late assignment (sickness, etc) please contact me directly or submit a medical certificate in the department secretary.

Plagiarism

Please read and strictly adhere to the [CIT Honesty, Plagiarism and Infringements Policy Related to Examinations and Assessments](#). Note that reports are **checked** against each other and against external web sources for plagiarism. Any suspected plagiarism will be treated seriously and may result in penalties and a zero grade.

Grading

The assignment is worth 35% of the overall mark for the module. Marks will be awarded based on the quality of the code and the results. In particular, I will be checking to see if you are handling and preprocessing data correctly, carrying out exploratory analysis to gain insights, correctly performing model implementation, and critically, documenting everything in a clear and concise way. The submitted code will also be checked to ensure that the work is your own.

Python environment setup for TensorFlow experiments

See below for how the Python environment is setup:

Installing Python Environment

Download Anaconda 5.2 (the Python 3.6 version) into Downloads

folder <https://www.anaconda.com/download/#linux>

Allow Anaconda to modify your .bashrc. This puts the

'conda' tool on the path.

Command:

bash Anaconda-latest-Linux-x86_64.sh

(First time only) Load Anaconda onto your path

so that it takes the place of the system Python

source ~/.bashrc

The following command should output "Python 3.6.x :: Anaconda

python3 -V

Using Ubuntu 16.04 or later (Linux Mint 18 or later)

Install jupyter notebook

<https://jupyter.org/install>

Create an environment called 'tensorflow' and install Tensorflow

conda create -n tensorflow python=3.6

Install spyder into the 'tensorflow' environment

conda install -n tensorflow spyder

Activate the TensorFlow Environment

source activate tensorflow

Tensorflow install.

pip install --ignore-installed --upgrade

<https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.10.1-cp36-cp36m->

[linux_x86_64.whl](https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.10.1-cp36-cp36m-linux_x86_64.whl)

Extra ML packages

conda install numba

conda install scikit-learn

conda install pandas

conda install keras

OCR packages

pip3 install opencv-python

pip3 install editdistance

```
# Fixes a bug with Spyder  
conda install tornado=4.5.3
```

```
##### end of install
```

When you want to develop with Python.

To use it:

1. Open a terminal
2. type "source activate tensorflow" (this activates a "tensorflow" environment)
3. type: jupyter notebook (or Spyder)

This should now give you Spyder running in the Anaconda / Python / Tensorflow environment.

(Tested with Ubuntu 18.04)