INFANT NUTIRION CLI TOOL

-Sourabh Kumar Janghel

This documents deals with the understanding of the solution and what components of the solution does what. This also talks about the model training part being used in the code and will also have a through comparison of results and what's working well and what is not.

Read this document after you are done deploying it, so that before running the solution on the data you understand the solution well and know how to use its different modes present through CLI.

Directory Structure-

a. **config.json** – This file contains the configuration required for the mongo DB connection and also the tesseract configuration being used to get text from the image. After several runs I decided on a generic configuration which works well on most of the image.
   a. As the official tesseract document also suggests to have a good quality image which can be read by human eye. I find many images not being readable enough but still tried to make a solution which is generic enough and works on most.

   Examples of bad image quality –
   mongoId("5e6a3befbf13490011565ef7")

b. **extractionCLI.py** – This is the main file that runs the whole program and calls the main components involved to extract and save the results.

c. **preprocessing.py** – This file runs various image processing functions on the image to make it more readable for the OCR and increase the accuracy. In this many years of working in ML field I haven't find a algorithm or steps that work on each image, but I tried to keep it generic enough so that it works on most of the samples. But there were quite a few that this pre-processing failed to accomplish anything and results where empty for ingredients and nutrients.

   Examples - 5e6a3b70bf13490011565c0c,
   5e6a3b99bf13490011565ca8,
   5e6a5c948a3d3004c8b009f7

d. **tesseract ocr.py** – This file handles the text extraction with the bounding box information. I used psm mode 11 which essentially captures all the text present in the image irrespective of the order and uses an oem mode of 2 which utilises the power of both legacy systems and LSTMs. Again this I tried keeping generic enough but still some of the images where failing text extraction.

Examples -5e6a3bc1bf13490011565db0, 5e6a3be4bf13490011565eab, 5e6a3befbf13490011565ef7, 5e6a3bf8bf13490011565f3d

e. **extractor.py** – This file contains the core logic that essentials handles the post-processing of the OCR results in computing the valid ingredients and nutrients. After several trials on the DATA provided I found that this solution works well on well focussed images of ingredients and nutrients and if both are separately given the results are very good. This algorithms or set o

f logics involves using a corpora that essentially contains all the nutrients, chemicals, additives involved in food items whic h I scrapped from various sources, so that the solution has a better context when it tries finding an ingredient or nutrient.

This extractor logic is also coupled with spell checker t o get correct name of the ingredients and nutrients along with E nglish grammar checking of the words in the contextual corpora d ictionary.

f. **prediction.py** – This is essentially a extra feature I added so to make the solution more accurate and also show that ML could be utilised in this task. This file takes an image and uses a ML trained Binary classifier to classify whether an image conta in information or not. The training file of this model along wi th the dataset that I created by manually annotating the images of being in either 0(absent information) or 1(present informati on) could be found in the model_training directory. Currently t his is a very basic version of the model which is slightly over fitted because of the data being imbalanced and could be retrai ned later to give better accuracy. Currently the accuracy I got is 83. I didn't had time to try various other techniques and fi nd the F1 score and AUC ROC scores to better understand the mod el.

Examples where the model fails to classify into the correct class- 5e6a 3b6bbf13490011565be8.

So to handle this I also created a rule based logic that also runs when a model incorrectly classifies and compares the OCR results with a set of ingredients and nutrients so that when we have a certain confidence that there are nutrients and ingredients present in the image the extra ction module starts extracting these information.

Some places my rule based logics of classifying was also failing. Example -5e6a3b9dbf13490011565cc4

I have stated each component and some examples to show where the soluti on is failing. I will in the next section share the examples where the solution is working well. These results contains some garbage which nee d more training and contextual information to be processed or removed.

In the next section I will try to focus more on the future development tis solution could have to better handle the current limitations and in crease the accuracy of the extraction.

Examples where the solution is working well –

- #5e6a3bacbf13490011565d22
- #5e6a3b6bbf13490011565be8
- #5e6a3bdfbf13490011565e87
- #5e6a5c248a3d3004c8b00736
- #5e6a5d5b8a3d3004c8b00f9e
- #5e6a5d4d8a3d3004c8b00f53
- #5e6a5cf78a3d3004c8b00cb5

- #5e6a5cbc8a3d3004c8b00b05
- #5e6a5c948a3d3004c8b009f7
- #5e6a5b9a8a3d3004c8b00428
- #5e6a5b808a3d3004c8b003a7
- #5e6a5b4d8a3d3004c8b00271
- #5e6a5b358a3d3004c8b001e8
- #5e6a43a2bf1349001156924e

LIMITATIONS OF THE SOLUTION

a. Image quality not being good results in bad pre-processing.
b. Doesn't handle skew and rotated text
c. Needs more specific pre-processing
d. More rules to be added to better handle rule-based classification
e. OCR configurations needs to be more specific
f. Model re-training needs to be done to handle imbalanced dataset.
g. More contextual vocabulary is need to make sense of the OCR extracted words.
h. Currently the solution only handles bigram word formation.

FUTURE DEVELOPMENT SCOPE

- Train a multiclass classifier with SMOTE techniques to handle imbalanced dataset where the classes could be –
  - No data
  - Only nutrients
  - Only ingredients
  - Both present
- According to the above trained model various logic as of now which is rule based could be improve to handle such scenarios. Currently the system handles them but is rule based so less confident and accurate.
- Trying not one but different pre-processing pipelines and then decide based on the ocr results which pipeline to consider for a given image.
- Improving the OCR extraction is very much dependent on image quality and the kind of pre-processing being done. This could also be given into different pipelines having different psm modes and oem modes for better extraction. We can also try to re-train the tesseract OCR with our data and check whether it gives good results.
- Adding n-grams that could essentially handle multiline ingredients, such as high-oleic safflower and high-oleic sunflower, which currently gets extracted as single token ingredients.
- DO more thorough post-processing to handle noise in the results.