2.1 Formula Recognition Using seq2seq Models - Preprocessing

August 5, 2019

1 Formula recognition using seq2seq models - Data Preprocessing

In this notebook, we describe the steps necessary to post process the data and put it in an easy to consume format.

The im2latex dataset consists of: > [A] total of ~100k formulas and images splitted into train, validation and test sets. Formulas were parsed from LaTeX sources provided here: http://www.cs.cornell.edu/projects/kddcup/datasets.html(originally from arXiv). Each image is a PNG image of fixed size. Formula is in black and rest of the image is transparent.

The image data provided is not terribly useful due to its large size and transparent background. Luckily, a person undertaking a project with this very same topic produced some helpful code. The code needed some trivial fixes to work with python 3.

To generate the dataset, run the code below. It renders each formula into a `.png` file, produces a list of formula to file mappings, an generates the vocabulary.

Note: Generating the full dataset takes several hours, and will run in the background. You will need to manually kill the python process if you wish to interrupt it. If you do this, get a and be prepared to wait.

In [2]:
```
# Custom src import
import sys
sys.path.append('../src/')
import utils
```

In [3]:
```
!pip install click imageio scipy
```

```
Requirement already satisfied: click in /Users/erikbeerepoot/.virtualenvs/ml-tf1/lib/python3.7/
Requirement already satisfied: imageio in /Users/erikbeerepoot/.virtualenvs/ml-tf1/lib/python3
Requirement already satisfied: scipy in /Users/erikbeerepoot/.virtualenvs/ml-tf1/lib/python3.7/
Requirement already satisfied: numpy in /Users/erikbeerepoot/.virtualenvs/ml-tf1/lib/python3.7/
Requirement already satisfied: pillow in /Users/erikbeerepoot/.virtualenvs/ml-tf1/lib/python3.7
You are using pip version 19.0.3, however version 19.2.1 is available.You should consider upgra
```

In [6]:
```
# Adapted from: https://github.com/guillaumegenthial/im2latex

import click
import json
```

```
from utils.data_generator import DataGenerator
from utils.text import build_vocab, write_vocab
from utils.image import build_images
from utils.general import Config

small_data = json.loads("""
{
    "export_name": "data.json",

    "dir_images_train": "../data/processed/formula/images/train/",
    "dir_images_test" : "../data/processed/formula/images/test/",
    "dir_images_val"  : "../data/processed/formula/images/validate/",

    "path_matching_train": "../data/processed/formula/images/train/train.matching.txt"
    "path_matching_val"  : "../data/processed/formula/images/test/test.matching.txt",
    "path_matching_test" : "../data/processed/formula/images/validate/val.matching.txt"

    "path_formulas_train": "../data/original/formula/small.formulas.norm.txt",
    "path_formulas_test" : "../data/original/formula/small.formulas.norm.txt",
    "path_formulas_val"  : "../data/original/formula/small.formulas.norm.txt",

    "max_iter"          : 20,
    "max_length_formula": 50,

    "bucket_train": false,
    "bucket_val": false,
    "bucket_test": false,

    "buckets": [
        [240, 100], [320, 80], [400, 80], [400, 100], [480, 80], [480, 100],
        [560, 80], [560, 100], [640, 80], [640, 100], [720, 80], [720, 100],
        [720, 120], [720, 200], [800, 100], [800, 320], [1000, 200],
        [1000, 400], [1200, 200], [1600, 200], [1600, 1600]
        ]
}
""")

# XXX - WARNING - XXX
# Processes the full dataset in the background. Takes a few hours.

full_data = json.loads("""
{
    "export_name": "data.json",

    "dir_images_train": "../data/processed/formula/images/train/",
    "dir_images_test" : "../data/processed/formula/images/test/",
    "dir_images_val"  : "../data/processed/formula/images/validation/",
```

```
    "path_matching_train": "../data/processed/formula/images/train.matching.txt",
    "path_matching_val"  : "../data/processed/formula/images/val.matching.txt",
    "path_matching_test" : "../data/processed/formula/images/test.matching.txt",

    "path_formulas_train": "../data/original/formula/images/train.formulas.norm.txt",
    "path_formulas_test" : "../data/original/formula/images/test.formulas.norm.txt",
    "path_formulas_val"  : "../data/original/formula/images/val.formulas.norm.txt",

    "bucket_train": false,
    "bucket_val": false,
    "bucket_test": false,

    "max_iter"          : null,
    "max_length_formula": 150,

    "buckets": [
        [240, 100], [320, 80], [400, 80], [400, 100], [480, 80], [480, 100],
        [560, 80], [560, 100], [640, 80], [640, 100], [720, 80], [720, 100],
        [720, 120], [720, 200], [800, 100], [800, 320], [1000, 200],
        [1000, 400], [1200, 200], [1600, 200], [1600, 1600]
        ]
}
""")

vocab = json.loads("""
{
        "export_name": "vocab.json",

    "unk": "_UNK",
    "pad": "_PAD",
    "end": "_END",
    "path_vocab": "../data/processed/formula/vocab.txt",
    "min_count_tok": 10
}
""")

def build_dataset(data, vocab):
    ''' Builds the im2latex dataset

    Arguments:
    data - configuration object describing data parameters (see example above)
    vocab - configuration object describing the vocabulary parmeters (see example abov
    '''
    print(" Building dataset... \n")
    data_config = Config(data)

    # datasets
```

```
            train_set = DataGenerator(
                path_formulas=data_config.path_formulas_train,
                dir_images=data_config.dir_images_train,
                path_matching=data_config.path_matching_train)
            test_set  = DataGenerator(
                path_formulas=data_config.path_formulas_test,
                dir_images=data_config.dir_images_test,
                path_matching=data_config.path_matching_test)
            val_set   = DataGenerator(
                path_formulas=data_config.path_formulas_val,
                dir_images=data_config.dir_images_val,
                path_matching=data_config.path_matching_val)

            # produce images and matching files
            train_set.build(buckets=data_config.buckets)
            test_set.build(buckets=data_config.buckets)
            val_set.build(buckets=data_config.buckets)

            # vocab
            print("\n")
            vocab_config = Config(vocab)
            vocab = build_vocab([train_set], min_count=vocab_config.min_count_tok)
            write_vocab(vocab, vocab_config.path_vocab)

        build_dataset(small_data, vocab)
        print("\n Success!  ")
```

```
 Building dataset...

Loaded 17 formulas from ../data/original/formula/small.formulas.norm.txt
Loaded 17 formulas from ../data/original/formula/small.formulas.norm.txt
Loaded 17 formulas from ../data/original/formula/small.formulas.norm.txt


Building vocab...
- done. 3/33 tokens added to vocab.
Writing vocab...
- done. 3 tokens

 Success!
```

Now that we have generated the dataset, we will put the labels in a form that is easier to work with. We also filter out any images that do not have a matching groundtruth label.

```
In [7]: import os

        ### Make sure our data is in order
```

```
data_base_dir = "../data"

original_data_path = data_base_dir + "/original/formula/"
processed_data_path = data_base_dir + "/processed/formula/"
pickle_data_path = data_base_dir + "/pickle/formula/"

assert os.path.exists(original_data_path), "Original data path does not exist."
```

It's convenient to pickle the labels so we can retieve them easily.

In [9]:
```
import pandas as pd
import numpy as np


with open(f"{original_data_path}train.formulas.norm.txt") as f:
    train_labels = np.array(f.readlines())

with open(f"{original_data_path}test.formulas.norm.txt") as f:
    test_labels = np.array(f.readlines())

with open(f"{original_data_path}val.formulas.norm.txt") as f:
    validation_labels = np.array(f.readlines())

train_matches = pd.read_csv(f"{processed_data_path}images/train/train.matching.txt", se
test_matches = pd.read_csv(f"{processed_data_path}images/test/test.matching.txt", sep=
validation_matches = pd.read_csv(f"{processed_data_path}images/validate/val.matching.t

print(f"Found {len(train_labels)} training labels.")
print(f"Found {len(train_matches)} training matches.")

print(f"Found {len(test_labels)} test labels.")
print(f"Found {len(test_matches)} test matches.")

print(f"Found {len(validation_labels)} validation labels.")
print(f"Found {len(validation_matches)} validation matches.")

# Get correct labels
train_labels = train_labels[[list(map(lambda f: f[1], train_matches))]]
test_labels = test_labels[[list(map(lambda f: f[1], test_matches))]]
validation_labels = validation_labels[[list(map(lambda f: f[1], validation_matches))]]

print(f"Kept {len(train_labels)} training labels.")
print(f"Kept {len(test_labels)} test labels.")
print(f"Kept {len(validation_labels)} validation labels.")
```

```
Found 76322 training labels.
Found 17 training matches.
Found 9444 test labels.
```

```
Found 17 test matches.
Found 8475 validation labels.
Found 17 validation matches.
Kept 17 training labels.
Kept 17 test labels.
Kept 17 validation labels.


/Users/erikbeerepoot/.virtualenvs/ml-tf1/lib/python3.7/site-packages/ipykernel_launcher.py:28:
/Users/erikbeerepoot/.virtualenvs/ml-tf1/lib/python3.7/site-packages/ipykernel_launcher.py:29:
/Users/erikbeerepoot/.virtualenvs/ml-tf1/lib/python3.7/site-packages/ipykernel_launcher.py:30:


In [ ]:
```