Shabby Pages*

Alexander Groleau[1] and Stefan Larson[2]

[1] Sparkfish, Austin TX, USA
agroleau@sparkfish.com
[2] SkySync, Ann Arbor MI, USA
slarson@skysync.com

Abstract. This paper presents the 'ShabbyPages' database, consisting of images of documents with realistic noise properties that result from standard office operations, such as printing, scanning, and faxing through old or dirty machines, degradation of ink over time, and handwritten markings. We designed this corpus to help train and evaluate machine learning methods – denoising, character recognition, and so on – with text documents. The dataset is publicly available online, alongside scripts to reproduce the dataset.

Keywords: optical character recognition · image processing · denoising · machine learning · image degradation · model training

1 Introduction

Scholars generally agree that humans invented writing around 5500 years ago in Mesopotamia, what is now present-day Iraq. Unfortunately, we're still far from achieving full literacy globally, with 16 percent of adults still unable to read and write fluently [1]. This situation is closely mirrored by developments in electronic language systems: we've been able to produce digital text for decades, but are only recently able to automatically read text back out of images of documents. Even now, this is not ubiquitous, with only carefully trained models able to do so under limited circumstances.

Inspired by the NoisyOffice dataset [2], we produced Shabby Pages as a way to help train, test, and calibrate computer vision machine learning algorithms designed for working with documents. We were particularly interested in producing a dataset more appropriate for training general denoising models, so we built and leveraged the Augraphy [3] document image augmentation tool to produce noise for these images.

2 Methodology

The formation of the dataset occurred gradually, in stages. A team of researchers scoured the open internet for "born-digital" documents - PDFs that were created entirely electronically, rather than scans of existing printed documents. 600

* Supported by Sparkfish LLC.

documents were collected for review, representing categories such as government press releases, corporate financial communiques, informational brochures, and many others.

The *pdftoppm* tool was used to separate each document into its constituent pages, converting these to PNGs in the process.

```
ls | grep pdf | time parallel pdftoppm {} 600dpi/{} -png -r 600
```

took 15 minutes and 7 seconds to process 6175 pages on a Ryzen 9 5950x, with the 1200 DPI version taking 44 minutes 49 seconds.

From there, we used Python's *cv2* library to convert each document's color channels to grayscale, and once color data was removed, we generated and applied an Augraphy pipeline. After augmenting the images, we fit each to a standard 8.5"x11" Letter document at various DPI levels, cropping to fit where necessary. There were several possible resizing methods, but we opted to use one which simulates using a document scanner to capture an image. Code for all of these processes is available on GitHub.

3 Shabby Pages - An Augraphy Project

In our investigations, we came across precious few sources of ground-truthed document images. We're releasing this corpus and the code we used to produce it, to aid the research community in making more. Training denoising models requires a large quantity of noisy data and the original clean sources, and producing this is exactly what the Augraphy library [3] was designed to facilitate. The Augraphy team has been hard at work for several months, improving the reliability, performance, and flexibility of the project, and we're proud that it's now mature enough to produce useful datasets.

Writing the augmentation pipeline was straightforward: we took the default Augraphy pipeline and parametrized all the augmentations within it, keeping the parameters at the top of a file for easy tuning. Supporting scripts were written to coordinate passing images through the Augraphy pipeline and saving them to new locations, deal with different image resolutions, and so on. It was then possible to reproduce the noising process, so a cycle of testing was conducted where we produced noised images from the pipeline, determined effects in the output we didn't want to include in the published set, and tweaked the pipeline accordingly. The Augraphy API enabled a tight feedback loop here, and made it possible to iteratively narrow in on the data we wanted.

The default pipeline has already been tuned over several months to produce realistic output, but the parameters required some careful tweaking to achieve the dataset we wanted. Even so, there wasn't a great deal of work to do here beyond fiddling with constants to add more sources of variation, and to reduce the

probability of certain augmentations being applied together. Driving Augraphy is largely a matter of developing some heuristics for unacceptable data and then telling Augraphy how to avoid producing that, so this part was largely minor edits to the build script and re-running the rendering job – a familiar workflow for practitioners and researchers in this field.

References

1. UNESCO News, https://en.unesco.org/news/literacy-all-remains-elusive-goal-new-unesco-data-shows. Last accessed 17 Feb 2022
2. UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets/NoisyOffice
3. Augraphy, An augmentation pipeline for rendering synthetic paper printing, faxing, scanning and copy machine processes, https://github.com/sparkfish/augraphy
4. Denoising Dirty Documents — Kaggle, https://www.kaggle.com/c/denoising-dirty-documents
5. DocCreator, https://doc-creator.labri.fr
6. Apricot, submodular optimization for machine learning, https://github.com/jmschrei/apricot