Automated Admissions Data Entry Clerk

Isaiah Chiu (‘19)

Westmont College, Department of Computer Science

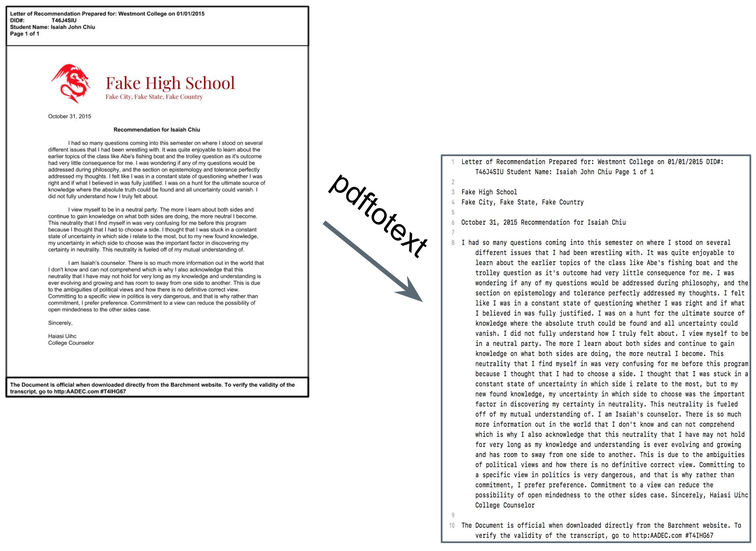
Abstract

The process of renaming application documents is a task that Westmont Admissions Data Entry Clerks spend countless hours on. The Automated Admissions Data Entry Clerk (AADEC) is a tool to automate the renaming process as it analyzes and classifies prospective students’ letters of recommendations, transcripts, school profiles, and so forth. The AADEC's data cleaning and extraction are handled by standard bash scripting, while the data analysis is done by the natural language processing (NLP) service from Amazon Web Services (AWS) which utilizes machine learning to classify the letters of rec. The entire workflow of the project will soon be compiled into a user friendly desktop application or easily executable file.

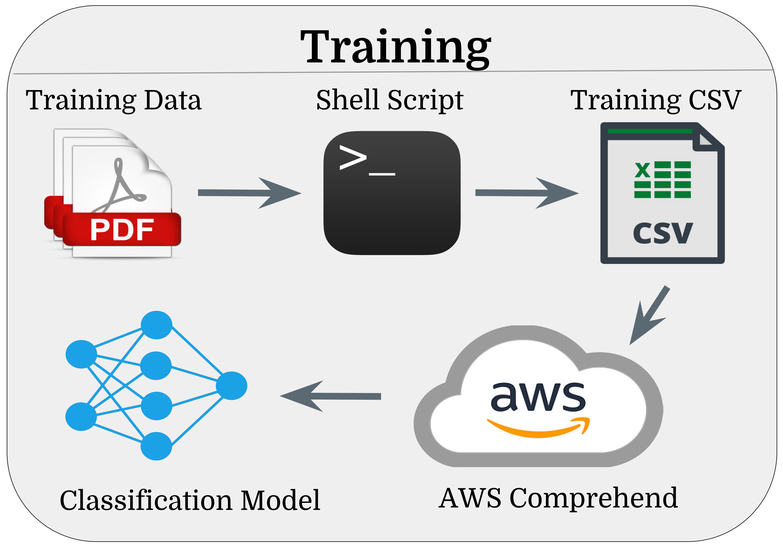
Data Preparation

During the data preparation phase, the following steps are done to ensure data cleanliness:

- Convert all pdf files into txt files using pdftotext
- Categorize all txt files: LORA, LORP, LORN
- Delete all excess text from files
- Build CSV file with classes in column 1 and the letter of recommendation in column 2
- Clean the CSV by checking for any null or “dirty” cells due to “pdftotext” errors

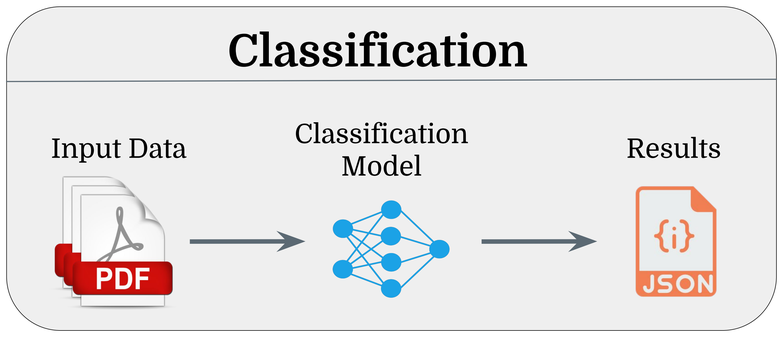


Workflow



Training Workflow

1. Training data gets taken into shell script as input
2. Shell script cleans all the data and outputs a CSV file
3. CSV file is uploaded to AWS S3
4. AWS Comprehend pulls CSV file from S3 bucket and builds classification model



Classification Workflow

1. Input data is uploaded to AWS S3
2. The classification model built in the training process runs a classification job on the input data
3. The classification model outputs a jsonl file containing the results of the analysis

Classification

The classification process has one primary step which is to use AWS Comprehend to build an NLP model based on the letters of recommendations provided by the user. This model classifies an input of files as either LORA, LORP, or LORN. Initially, the performance statistics for the model were not very impressive and I assumed it was because of the disproportionate amount of LORAs compared to the LORPs and LORNs in the dataset. I tried to undersample the LORAs in an attempt to work around the imbalance and ended up with the same results. I finally decided to build a binary classification model by merging the LORP and LORN classes together. Here are the improvements to the model: [Accuracy, +1.72%], [Precision, +35.32%], [Recall, +20.19%]

Classifier Performance

[Accuracy: 90.8%] [Precision: 91.04%] [Recall: 70.25%]

Results

Sample Classification Job Output:

```
{
"File": "Chiu_FY_5345239_OE_050884_Isaiah.txt",
   "Line": "0",
   "Classes":
      [
         {"Name": "LORP", "Score": 0.923},
         {"Name": "LORA", "Score": 0.077}
      ]
}
```

- When the model classifies a LORA, it has an extremely high probability that it will predict it correctly
- When the model classifies a LORP, it will correctly predict the document approximately 50% of the time

		Predicted Class	
		LORA (P)	LORP (N)
Actual Class	LORA (P)	96	2
	LORP (N)	20	14

Next Steps:

- Implement data extraction and document renaming
- Create workflow that can be easily used by admissions