PDF Title Extraction

The task is to identify important sections (titles) from styled pdf documents, as they represent anchor points for the extraction of various other important information. In the attached CSV files (split into training and testing), you will have access to a limited set of features from sample documents.

The structure of these CSVs is as follows:
• Each line represents information about one particular block of text from the document. This information contains:
  o Text: the raw text of the section as interpreted by the OCR;
  o IsBold: is the section bold or not; o IsItalic: is the section italic or not;
  o IsUnderlined: is the section underlined or not;
  o Left: the left coordinate on the page;
  o Right: the right coordinate on the page;
  o Top: the top coordinate on the page;
  o Bottom: the bottom coordinate on the page; and
  o Label: a label to state is the section represent a title (1) or normal text (0).

You will also find some sample pdfs files to give you a better idea of the documents you are dealing with (titles are highlighted by red rectangles).

Your task is to build a machine learning model that is capable of classifying sections as titles and nontitles. You are free to use any machine learning model you are comfortable with (deep learning models are preferable but not mandatory).

We expect you to submit the following as part of your solution:
• A fully working python project to achieve the above task (with a setup script).
• A report stating the following:
  o The accuracy achieved against the test set.
  o A brief explanation of your solution including reasons for your model of choice.
  o A description of any improvements you would have made given extra time.
  o Please also specify the amount of time you have spent designing your solution. We intend to run your solution, so please give us an idea of setup, train and run time

Dataset

Two datasets have been provided. train_sections_data.csv and test_sections_data.csv. Here is some records from the test dataset.

Text	IsBold	IsItalic	IsUnderlined	Left	Right	Top	Bottom	FontType	Label
Employee Involvement	TRUE	FALSE	FALSE	77.9	175.6	425.3	434.2	New Times Roman	1
Income Taxes	TRUE	TRUE	FALSE	33.1	79.6	284.8	290.6	New Times Roman	1
•	FALSE	FALSE	FALSE	77.9	81.3	346.1	355	New Times Roman	0
the Audit & Remuneration Committee	FALSE	TRUE	FALSE	124.8	247.3	380.3	385.9	New Times Roman	0

Data Exploration

After combing through the dataset I noticed a couple of things.

Data Imbalance

The dataset is imbalanced. There are 3695 samples labelled ‘Title’, out of a total 14215 samples in the training set (26%).

Text Formatting
There are three text formatting features in the dataset IsBold, IsItalic and IsUnderlined. Intuitively I thought that IsBold and IsUnderlined would be extremely useful features. Titles are often underlined and in bold.

Location
The location of the text in the pdf is another set of features that will provide our model with large amounts of information. Titles will often be positioned centrally in a pdf, often toward the top of the document.

FontType
This feature is totally useless. Every one of the samples in the dataset had the FontType ‘New Times Roman’

Solution

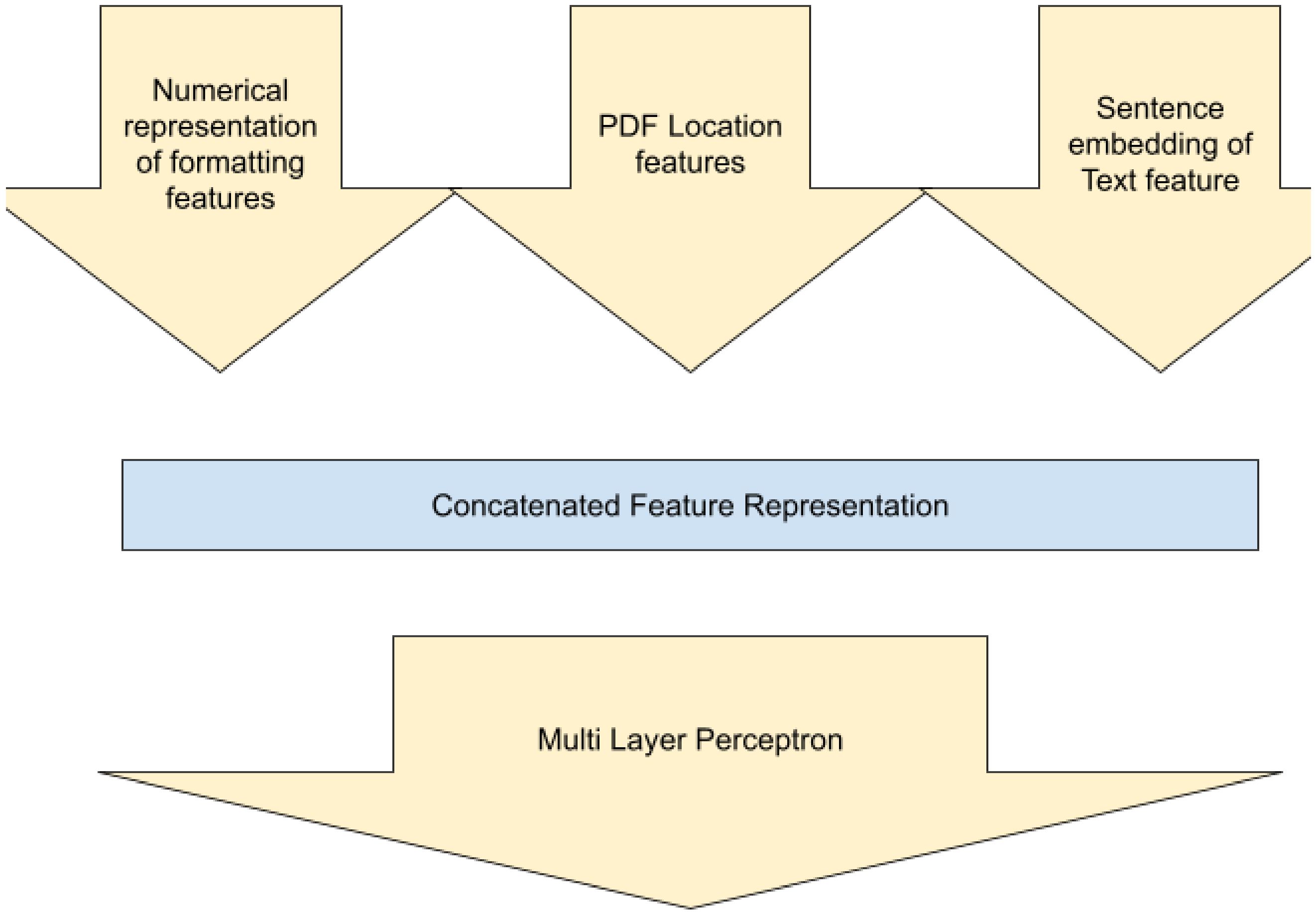
Time to complete

In total I spent about 5 hours working on my solution on and off

Model Architecture

o A brief explanation of your solution including reasons for your model of choice.

I went for an ensemble approach combining three different sets of features.


Numerical representation of formatting features
PDF Location features
Sentence embedding of Text feature
Concatenated Feature Representation
Multi Layer Perceptron

Formatting Features

The dataset contained columns related to the formatting used in the sampel text. These were boolean columns stating whether the formatting had been applied or not. II converted these features in to numerical representations. 1 being True, 0 being False.

Location Features

The dataset contained columns related to the location of sample text on the pdf it is a apart of. I took these features as they were and concatenated them to the rest of the numerical features.

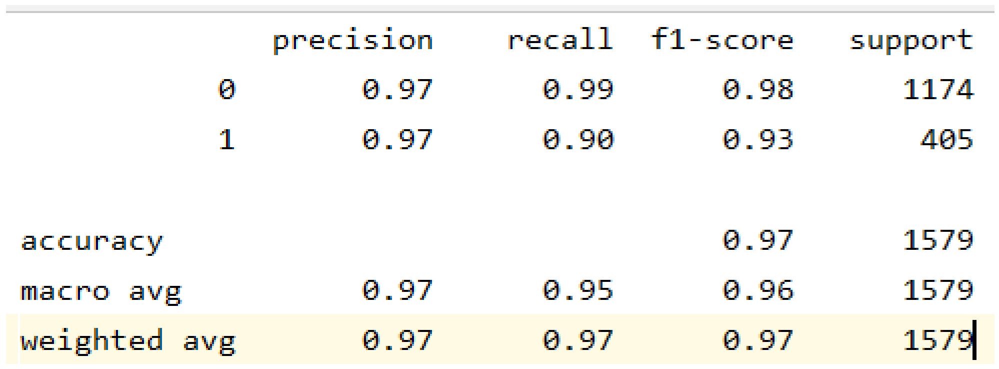
Sentence Embeddings

Sentence embeddings capture the semantic meaning of text. This is crucial for this task as we are dealing with titles in PDF documents. Titles often convey specific semantic importance, and sentence embeddings help in understanding the contextual meaning of these titles. This is especially important given the small amount of samples in the dataset (26%), contextual embeddings from pre trained language models com pre loaded with general language understanding, which our model can leverage to learn the linguistic features that indicate that text is a ‘Title’. This boost in understanding will mean the model can learn quicker, and from fewer samples then a non-pretrained model.

I used the ‘multi-qa-MiniLM-L6-cos-v1’ model from sentence-transformers as my text embedder.It maps sentences & paragraphs to a 384 dimensional dense vector space and was designed for semantic search. It has been trained on 215M (question, answer) pairs from diverse sources. This means it has learned to generate robust representations for a wide range of textual content. This robustness is valuable when dealing with the varied writing styles and formats that might be present in a PDF document.

Training Results

Achieved an accuracy of 97%. The F1 score, Precision, Recall and accuracy achieved against the test set below:



```
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      1174
           1       0.97      0.90      0.93       405

    accuracy                           0.97      1579
   macro avg       0.97      0.95      0.96      1579
weighted avg       0.97      0.97      0.97      1579
```

Future Improvements

Future improvements:

- LLM experiments: Would be interesting to see how an LLM fares with few shot learning
- Sentence embedding improvements
  - Using a Larger embedding model may increase performance
    - More expressive due to more parameters
  - Experimenting with he max_length of embeddings
- Location feature normalisation
- Code Refactoring
  - Write all docstrings in the project following standard python formatting
  - Id use a linter to refactor code in standard python guidelines (would use black linter)
- Efficiency improvements
  - I could have encoded all of the text features records at one time using sentence transformers, instead of record by record. This would save alot of time and compute

Running the project

Run time

It takes 10 minutes to train and evaluate a model via cpu on a GPU with 16gb of RAM.

Instructions

If you do not already have the project directory saved to your development machine you can clone it via this link: https://github.com/ajbanks/pdf-title-extraction.git.

Prerequisites:

- Run all commands from the project root directory
- Use Python 3.11.5.
- Ensure that the train and test csv files are in a directory called 'data' wherever you are running the package
  ```
  data/train_sections_data.csv
  data/test_sections_data.csv
  ```

1. Install project

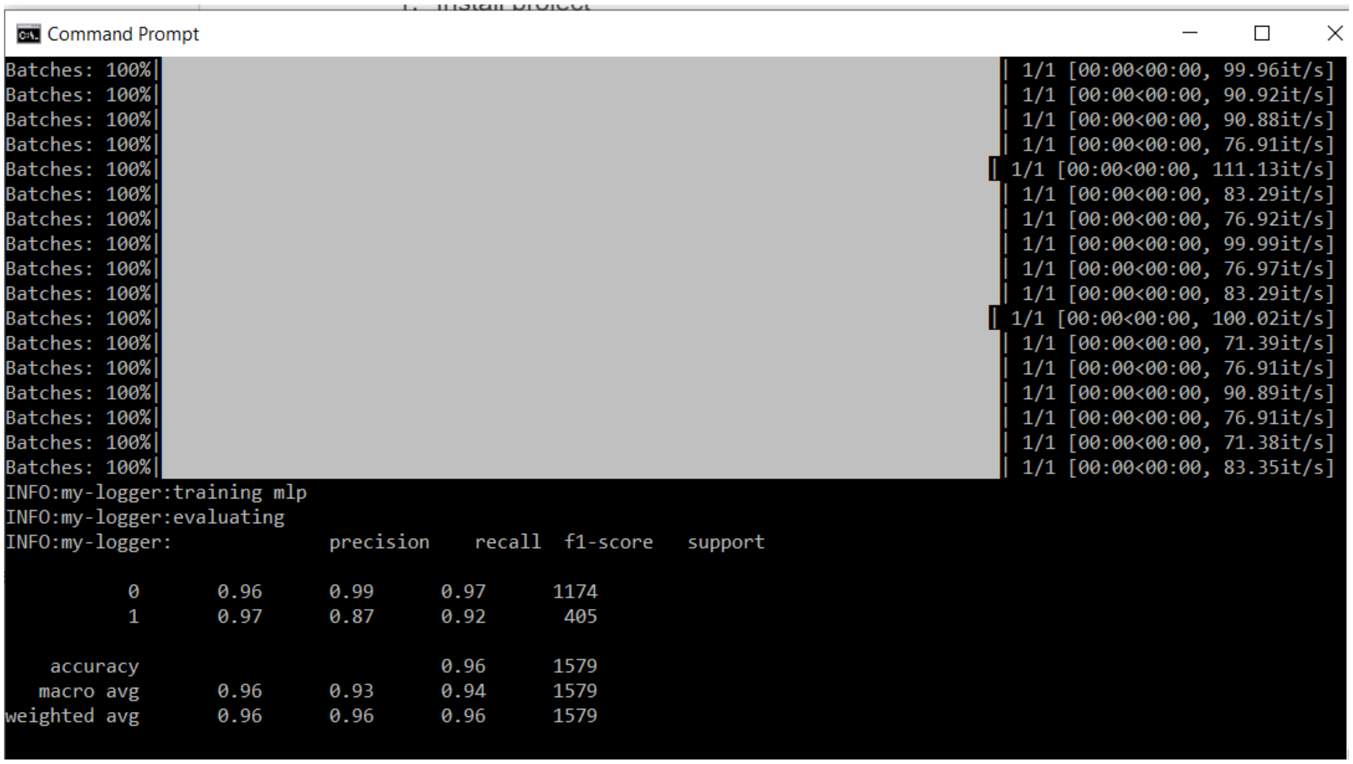
   ```
   pip install .
   ```

2. Code to first run training and evaluation, then print results to cmd line (assumes data is in a directory in the same location the script is run 'data/')

   ```
   from titleextraction.title_detection_model import Title_Detection_Model

   pdf_title_detection_model = Title_Detection_Model()

   pdf_title_detection_model.end_to_end_training()
   ```


Command Prompt

```
Batches: 100%|                                        | 1/1 [00:00<00:00, 99.96it/s]
Batches: 100%|                                        | 1/1 [00:00<00:00, 90.92it/s]
Batches: 100%|                                        | 1/1 [00:00<00:00, 90.88it/s]
Batches: 100%|                                        | 1/1 [00:00<00:00, 76.91it/s]
Batches: 100%|                                       | 1/1 [00:00<00:00, 111.13it/s]
Batches: 100%|                                        | 1/1 [00:00<00:00, 83.29it/s]
Batches: 100%|                                        | 1/1 [00:00<00:00, 76.92it/s]
Batches: 100%|                                        | 1/1 [00:00<00:00, 99.99it/s]
Batches: 100%|                                        | 1/1 [00:00<00:00, 76.97it/s]
Batches: 100%|                                        | 1/1 [00:00<00:00, 83.29it/s]
Batches: 100%|                                       | 1/1 [00:00<00:00, 100.02it/s]
Batches: 100%|                                        | 1/1 [00:00<00:00, 71.39it/s]
Batches: 100%|                                        | 1/1 [00:00<00:00, 76.91it/s]
Batches: 100%|                                        | 1/1 [00:00<00:00, 90.89it/s]
Batches: 100%|                                        | 1/1 [00:00<00:00, 76.91it/s]
Batches: 100%|                                        | 1/1 [00:00<00:00, 71.38it/s]
Batches: 100%|                                        | 1/1 [00:00<00:00, 83.35it/s]
INFO:my-logger:training mlp
INFO:my-logger:evaluating
INFO:my-logger:              precision    recall  f1-score   support

           0       0.96      0.99      0.97      1174
           1       0.97      0.87      0.92       405

    accuracy                           0.96      1579
   macro avg       0.96      0.93      0.94      1579
weighted avg       0.96      0.96      0.96      1579
```