Documentation

OBJECTIVE: TO EXTRACT STRUCTURED MENU FROM PDF DOCUMENTS

The Program has following parts:

1)Extracting text from pdf’s using extractor:

Using an open source software called Apache Pdf box and extracted the entire text of different pdf files kept in a source directory to a single text file.

2) Finding item value pairs:

Extract an item and its corresponding cost from pdf file using structured properties that item and price come as pairs which are close to each other in the text file and using trained data about what has the property to be an item .

Procedure applied:

1) Using apache pdf box I am printing the meta information of the pdf to the console
2) Stripper.writetext method extracts entire data from a single pdf and it depends on the internal layout of the pdf file so the text may be extracted row wise or column wise depending on the proximity of the text in pdf.
3) Running the same code over all the files in the directory extracting all the pdf ‘s in the directory into a single file.
4) Now I have taken some sample pdf’s and extracted there text as said above and I manually identified items and produced training data.
5) Now I sent the training data to an NLP open source software APACHE OPEN NLP and there specifying my model to an unigram model and additional feature generators necessary and generated a training model which has the ability to identify an item
6) Then for any other pdf’s using steps 1-3 extracted data to a text file and using the trained model I have I identify items and store it into a text file.
7) Now Using regular expressions which matches to prices like $20 and Rs. 50 etc. identified all prices .
8) Then found words which were at close proximity to these values and took them as item here close proximity means the data from the pdf to its nearest new line.
9) Then verified whether trained model actually identified the item if not I would add the item to the training data
10) Now price need not start with a symbol or Rs. It could have been an pure number in which our training model comes to help Ex: 100 which has the possibility of being a price or not being a price like it can be a phone no or pin code
11) So I must verify whether it is actually an item or not so I verify from the text file containing whether it is really an item and add it otherwise I ask user whether it is an item or not and if answer is yes I add it to both training data and item if he says no I would leave the item.

Running the code:

1) Now use the Extract.java method and extract the entire information when you provide the directory where pdfs are present and output file where you want to be extracted.
2) Now generate a bin file containing the trained model using SentenceTrainModel 's createTrain method if the bin file is absent or there is any modification to the trained data otherwise this step is unnecessary.
3) Now using the bin file generated in the step 2 and input the text file from step 1 and using the SentencePossibilityGenerator's createCheck method create a new file which has the possibility to be an item in your input file you can skip this step if such a file is already available.
4) Now using the Steps 3 text file and input file the output is written to another file as item= , Value= and also some items which are not identified into the Training model both in a automated and manual way Now the training model is changed so the next time you use the project it is better to start from step 1 for better results .
5) Here all file paths are hard coded to run you must atleast have directory of pdf and trained file and training file must be place correctly.

Explanation of the Code:

1)Extract.java :

This contains the stripper method which will strip the pdfs text into a file and it has also many methods which provide meta information to the console which are self explanatory.

2)SentenceDetectTrain.java:

This contains the train method which trains from the from the trained data the trained data must be of this format <START:item> Chicken 65 <END> these should be spaces properly maintained also as shown in the example otherwise the training data is not accepted.
The no of iterations currently provided is 100 which can be increased to large number for better results.

3)Item PossibilityGenarator.Java:

Now the input from the pdf 's using Extract.Java is provided as a input to the this and we should break into a individual words and using the bin file generated in the step above using nameFinder.find method we find where the items are spanned into a span array and now outputing the data into text file

4)ItemValueGenarator.Java:

1)Now in this method I have a taken a string s1 where I have kept the entire text and removing the numbering at the start of line indicating line number if present from each line like 67) chicken 65 $20 Is taken as chicken 65 $20 removing such a formatting but it can be a

Also 2 pcs of chicken or so on which case it is not any line formatter in that case where it is a pure number I check what follows has something like piece , pcs or l or ml and in that case converting the number to English word representation of the number for the digit.
2) 3 regular expressions are written to identify the prices and the | method is used to let it detect by any one of these methods.

3) Now i have taken the entire file generated in step 3 to string s2 and splitted the entire S1 at '/n' and I am taking each line and matching with regex and taking a count of no of matches and if no match is gound leaving the sentence

4)If a match is found and there is exactly one match and text preceding it in that case taking charecters till the nearest new line as item and remaining as value which is the simplest case.

5)During Extract.java row wise extraction might not have happened and column wise extraction might have happened   in which case we get each item in a newline followed by each price in a new line in which case what I do is check if the regex match spans the entire line in such a case I create an array and start noting the value of prices into the array and when such a match fails it takes each line and corresponds to correct entry which happens in cnt>0 and here the variable lb is used because to check end is reached and some items Still some items are printed to make it come into for loop again.

6)Now there might be a case of Chicken Tikka /mutton Tikka $6.5/$7.5 in this case we must match ChickenTikka $6.5 and MuttonTikka $7.5 So if count>1 and I am trying to split what follows at the '/' symbol into arr and if arr.length=count just match the corresponding the arr to value just moving forward the match

7)Now if such a thing does not happen and arr.length>count like Chicken Tikka /mutton Tikka to $7.5 then printing the output as item=ChickenTikka/muttonTikka and value= $6.5

8) Now if arr.length<count it can be of a case of Chicken 65 $2.4 in which count will be 2 Now I match it with a regular expression containing symbol involved in it i.e arr[0] and arr[1] and if this count is less than original count it must have been as specified above in which case I just append the first match to string and take the value as next match so item=Chicken 65 value =$5

9)Now in the same arr.lenght<count with Chicken 65 100 and in such a case I would check if the next match extends to the end and output is similar to case 8

10)Now in the same arr.length<count we have an input Chicken $5 $10 $15 with the headings above I would move one line above this and the heading would be something like Small Medium Large or have more than 1 word if there is more than 1 1word I assumed that

Same no of words would be there in each so I used that technique to find each corresponding entry Then I would match each with its corresponding heading like Chicken Small $5, Chicken Medium $10, Chicken Large $ 15

11)Now there could be more than one item pairs in the same line like
Chicken $5 Mutton $10 So In that case arr.length<count and match does nor really extend upto end in which case I match first string with first value and second string with second value and so on .

Continuation of the project:

1)The project was mainly designed for items of hotel menu this can be extended further by just training manually for other data items pdf using the same code and the same code can be extended to Ms word documents we can extract data from them using an open source software apche poi and html document can be extracted based on the html tags without using this method.

2)The code can be extended to make it more fault tolerant as exception occurs at some point it should not stop program flow stop.

3)The code must be extended to allow concurrent programming techniques as it takes a large amount of time to run the code.