using IronOcr;

using System.Drawing; //for image export

// We can delve deep into OCR results as an object model of

// Pages, Barcodes, Paragraphs, Lines, Words and Characters

// This allows us to explore, export and draw OCR content using other APIs/

var Ocr = new IronTesseract();

Ocr.Configuration.EngineMode = TesseractEngineMode.TesseractAndLstm;

Ocr.Configuration.ReadBarCodes = true;

using (var Input = new OcrInput(@"img\Potter.tiff"))

{

OcrResult Result = Ocr.Read(Input);

foreach (var Page in Result.Pages)

{

// Page object

int PageNumber = Page.PageNumber;

string PageText = Page.Text;

int PageWordCount = Page.WordCount;

// null if we don't set Ocr.Configuration.ReadBarCodes = true;

OcrResult.Barcode[] Barcodes = Page.Barcodes;

System.Drawing.Bitmap PageImage = Page.ToBitmap(Input);

int PageWidth = Page.Width;

int PageHeight = Page.Height;

foreach (var Paragraph in Page.Paragraphs)

{

// Pages -> Paragraphs

int ParagraphNumber = Paragraph.ParagraphNumber;

String ParagraphText = Paragraph.Text;

System.Drawing.Bitmap ParagraphImage = Paragraph.ToBitmap(Input);

int ParagraphX\_location = Paragraph.X;

int ParagraphY\_location = Paragraph.Y;

int ParagraphWidth = Paragraph.Width;

int ParagraphHeight = Paragraph.Height;

double ParagraphOcrAccuracy = Paragraph.Confidence;

OcrResult.TextFlow paragrapthText\_direction = Paragraph.TextDirection;

foreach (var Line in Paragraph.Lines)

{

// Pages -> Paragraphs -> Lines

int LineNumber = Line.LineNumber;

String LineText = Line.Text;

System.Drawing.Bitmap LineImage = Line.ToBitmap(Input); ;

int LineX\_location = Line.X;

int LineY\_location = Line.Y;

int LineWidth = Line.Width;

int LineHeight = Line.Height;

double LineOcrAccuracy = Line.Confidence;

double LineSkew = Line.BaselineAngle;

double LineOffset = Line.BaselineOffset;

foreach (var Word in Line.Words)

{

// Pages -> Paragraphs -> Lines -> Words

int WordNumber = Word.WordNumber;

String WordText = Word.Text;

System.Drawing.Image WordImage = Word.ToBitmap(Input);

int WordX\_location = Word.X;

int WordY\_location = Word.Y;

int WordWidth = Word.Width;

int WordHeight = Word.Height;

double WordOcrAccuracy = Word.Confidence;

if (Word.Font != null)

{

// Word.Font is only set when using Tesseract Engine Modes rather than LTSM

String FontName = Word.Font.FontName;

double FontSize = Word.Font.FontSize;

bool IsBold = Word.Font.IsBold;

bool IsFixedWidth = Word.Font.IsFixedWidth;

bool IsItalic = Word.Font.IsItalic;

bool IsSerif = Word.Font.IsSerif;

bool IsUnderLined = Word.Font.IsUnderlined;

bool IsFancy = Word.Font.IsCaligraphic;

}

foreach (var Character in Word.Characters)

{

// Pages -> Paragraphs -> Lines -> Words -> Characters

int CharacterNumber = Character.CharacterNumber;

String CharacterText = Character.Text;

System.Drawing.Bitmap CharacterImage = Character.ToBitmap(Input);

int CharacterX\_location = Character.X;

int CharacterY\_location = Character.Y;

int CharacterWidth = Character.Width;

int CharacterHeight = Character.Height;

double CharacterOcrAccuracy = Character.Confidence;

// Output alternative symbols choices and their probability.

// Very useful for spell checking

OcrResult.Choice[] Choices = Character.Choices;

}

}

}

}

}

}