INFO: AbbyyEngine - Overview

NEW as of 10.7, (and up through 11.2) Atalasoft has added AbbyyEngine for OCR.

This new AbbyyEngine is a high quality, fast, and accurate OCR engine with ICR capabilities.

NOTICE OF END OF LIFE:

As of 11.3, AbbyyEngine is no longer offered. Existing AbbyyEngine customers will be given OmniPageEngine licensing going forward.

Please see: INFO:OmniPageEngine - Overview

Requirements

The AbbyyEngine requires a license for Atalasoft OCR with the Abbyy Add-on.

Additionally, we do not include the Abbyy OCR Resources with the DotImage download as the resources download is around 400 MiB in size.

Download Abbyy 11.2 OCR resources here

Download Abbyy 11.1 OCR resources here

Download Abbyy 11.0 OCR resources here

Download Abbyy 10.7 OCR resources here

After downloading unzip them to: C:\Program Files (x86)\Atalasoft\DotImage 11.1\bin\OcrResources\Abbyy

If targeting .NET 3.5, you must also include Interop.FREngine.dll form the Abbyy engine bin directory in your project bin directory (for .NET 4.0, and up this step is not needed)

NOTE: AbbyyEngine does not support .NET 2.0 or 3.0 - you must target .NET 3.5 or higher

Your project will need to reference Atalasoft.dotImage.Ocr.dll and

INFO: AbbyyEngine - Overview

Atalasoft.dotImage.Ocr.Abbyy.dll

In order to use the AbbyyEngine you need to use an AbbyyLoader in your code (more below)

Getting Started

1) Download the Abbyy OCR resources and unzip them to: C:\Program Files (x86)\Atalasoft\DotImage 10.7\bin\OcrResources\Abbyy

2) Ensure you have either a valid evaluation license for Atalasoft OCR with Abbyy or you have a paid OCR license that includes the Abbyy Add-on

3) In your project, add references to Atalasoft.dotImage.Ocr.dll and Atalasoft.dotImage.Orc.Abbyy.dll (in addition to the normal Atalasoft references such as Atalasoft.dotImge.dll, Atalasoft.dotImage.lib.dll, and Atalasoft.Shared.dll)

4) If targeting .NET framework 3.5, make sure you add Interop.FREngine.dll directly to your bin folder of your application. (please see the relevant section below). You can do this by adding a reference to the dll and ensuring that "Copy Local" is set to true

5) In a static constructor for your class you will need to call

```
string ocrResourcePath= @"C:\Program Files (x86)\Atalasoft\DotImage 10.7\bin\OcrResources\Abbyy";

AbbyyLoader loader = new AbbyyLoader(ocrResourcePath);
```

NOTE: this instruction assumes you used the default location back in step 1. When deploying, you'll need to point this loader to the location where the Abbyy OcrResources end up being deployed on the client system

6) You should now be able to instantiate an AbbyyEngine object, initialize it, and start using it.

More About Interop.FREngine.dll

INFO: AbbyyEngine - Overview

This file concerns customers using .NET Framework 3.5.

Users of .NET Framework 4 need not concern themselves with this file.

For any code compiled under Framework 3.5; this .dll file must be included in the same folder along with any others (such as the Atalasoft.dotImage.Ocr.Abbyy.dll file) that you're using during development and then the deployment of your application.

The easiest way to go about this is to add it as a reference to your .NET project in Visual Studio and make certain that the Copy Local property of the reference is set to true. This should automatically ensure that it's copied to your bin or output folder.

Following that, you should manually ensure that this .dll is included in your deployed application along with all the others.

AbbyyEngine Available Preprocessing Options

Our OcrEngine class has a property that allows you to determine which OcrPreprocessingOptions are supported by a given engine. Here are the results for AbbyyEngine:

AvailablePreprocessingOptions:

AutoRotate: True

Deskew: True

Despeckle: True

FlipLeftRight: False

Invert: True

ToBilevel: True

ICR With AbbyyEngine

In addition to the standard functionality, the AbbyyEngine also supports some extra features that most other engines do not. First and foremost – ICR (Intelligent Character Recognition), used for recognizing printed handwritten text, including such when individual characters are enclosed by frames and borders.

In order to provide support for ICR, a special method has been added specifically to the AbbyyEngine class:

```
Recognize(AtalaImage image, List<AbbyyTextRegion> abbyyRegionList)
```

The method above takes an AtalaImage as a parameter, as usual, and it also takes a List consisting of AbbyyTextRegion objects. AbbyyTextRegion is an abstract class, which itself extends OcrTextRegion (see section 5.1), and which has two concrete implementations:

```
AbbyyOcrTextRegion

AbbyyIcrTextRegion
```

What this all means is that it is possible to create a list consisting of a mix of these objects (or a list consisting of just one type or the other type); and pass them into a Recognition process along with the image (AtalaImage) that they pertain to. The idea is that an object of one of these types holds information on its own location in the image (x & y co-ordinates, width and height), as well as optionally, the orientation of the text marked by it in relation to the page; all of this info is used by the FineReader Engine for configuration of the Recognition process. The type of object itself tells the ABBYY FineReader what operation to perform on the corresponding region in the image; i.e. OCR recognition if it’s an AbbyyOcrTextRegion, or ICR recognition if it’s an AbbyyIcrTextRegion.

Construction of an AbbyyOcrTextRegion can be performed via one of two constructors:

```
AbbyyOcrTextRegion(Rectangle bounds)

AbbyyOcrTextRegion(Rectangle bounds, OcrTextRotation rotation)
```

The Rectangle bounds parameter referenced in both of these constructors is meant to designate the location of the region in the image. The OcrTextRotation rotation parameter designates the orientation of the text in relation to the top of the document. If this parameter is not given (i.e. the simpler constructor is called), then the text referred to by this AbbyyOcrTextRegion is assumed to be at 0 degrees in relation to top.

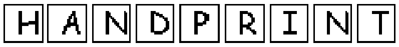
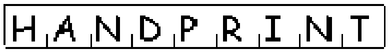
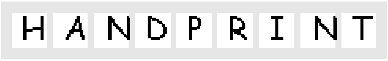
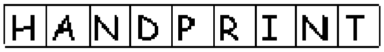
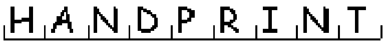
Construction of an AbbyyIcrTextRegion can also be performed via one of two constructors, corresponding exactly to the constructors of the AbbyyOcrTextRegion class:

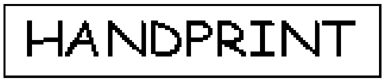
```
AbbyyIcrTextRegion(Rectangle bounds)

AbbyyIcrTextRegion(Rectangle bounds, OcrTextRotation rotation)
```

However, the AbbyyIcrTextRegion class also houses 2 additional client-settable properties that the AbbyyOcrTextRegion does not:

TextBorder refers to the type of border that handwritten printed text is sometimes found enclosed in within documents. This property is of type AbbyyTextBorder and has one of several values, corresponding to the values, as follows:

Border Type	Description	Example Image
`AbbyyTextBorder.CharBoxSeries`	This value specifies that the field where the text is located is a set of separate boxes.	
`AbbyyTextBorder.CombInFrame`	This value specifies that the field where the text is located is a comb and that this comb is also the bottom line of a frame.	
`AbbyyTextBorder.GrayBoxes`	This value specifies that the text is located in white fields on a gray background.	
`AbbyyTextBorder.PartitionedFrame`	This value specifies that the field where the text is located is a frame and this frame is split by vertical lines.	
`AbbyyTextBorder.SimpleComb`	This value specifies that the field	

	where the text is located is a comb.	
AbbyyTextBorder.SimpleText	This value denotes the plain text.	
AbbyyTextBorder.TextInFrame	This value specifies that the text is enclosed in a frame.	
AbbyyTextBorder.UnderlinedText	This value specifies that the text is underlined.	

CellCount is the 2nd user-settable property of the AbbyyIcrTextRegion class. From an examination of the border types shown above - it becomes apparent that certain types of borders delimit individual characters, while others do not. This is what this property pertains to. For the relevant types of borders, CellCount should be set to the amount of cells present in the region as a result of borders within the AbbyyIcrTextRegion delimiting individual characters.

If the TextBorder border and CellCount properties are not explicitly set by the client given; then the former border defaults to the value of AbbyyTextBorder.SimpleText, the later defaults to 1.

OK, we’ve covered a lot of ground in this section. So here’s an example section of code, to give a better idea as to how this might all look in practice:

```
... // declare and initialize AbbyyEngine, scan or load in AtalaImage


List<AbbyyTextRegion> regionsList = new List<AbbyyTextRegion>();

regionsList.Add(new AbbyyOcrTextRegion(new Rectangle(200, 657, 200, 30)));
regionsList.Add(new AbbyyOcrTextRegion(new Rectangle(785, 344, 100, 200),
OcrTextRotation.Clockwise90));
```

```
AbbyyIcrTextRegion icrRegion = new AbbyyIcrTextRegion(new Rectangle(401, 1148, 160, 45));

icrRegion.TextBorder = AbbyyTextBorder.PartitionedFrame;

icrRegion.CellCount = 8;

regionsList.Add(icrRegion);


abbyyEngine.Recognize(image, regionsList);
```

Finally, it should be noted that there is another way to launch ICR recognition; particularly useful if it is not Recognition that is desired, but Translation.

This approach essentially entails the creation of a custom OcrPageLocationEventHandler by the client, and registering it to the PageLocation event of the AbbyyEngine instance (it should be registered before Recognition or Translation is launched).

This handler should retrieve the collection (OcrRegionCollection) of recognized regions outputted by the engine from the RegionsIn property of the OcrPageLocationEventArgs object returned from the event (or create a new collection if this property is null).

After retrieving this collection, the client can add new objects of type OcrTextRegion, whose TextKind properties should be set to type OcrTextKind.HandPrint; and the bounds of which should correspond to the locations that the client is interested in performing ICR in.

After adding these new regions to the collection, the collection should be assigned to the RegionsIn property of the OcrPageLocationEventArgs object.

Example code is shown below:

```
    tr.TextKind = OcrTextKind.HandPrint;

    coll.Add(tr);

    e.RegionsOut = coll;

};
```

Supported Languages

The list of supported languages is mostly given by the following link:

http://www.abbyy.com/support/finereader/11/rl/

However there are a number of exclusions and exceptions from that list; including all Cyrillic-script based languages (such as Russian, Bulgarian, etc…), and a number of the more obscure/rare languages that are not supported in .NET with corresponding CultureInfo identities.

For your convenience, here is a current (As of 11.0.0.6 (March 2018) list of Supported Languages from our Abbyy OCR engine

LANGUAGES	Icelandic	Rwanda
Afrikaans	Ido	Sami, Northern (Norway)
Albanian	Indonesian	Samoan
Arabic	Interlingua	Scottish Gaelic
Armenian	Irish (Ireland)	Serbian
Armenian (Grabar)	isiXhosa (South Africa)	Shona
Armenian (Western)	isiZulu (South Africa)	Slovak
Aymara	Italian	Slovenian
Azeri	Japanese	Somali
Basque	Japanese (Modern)	Sotho
Bemba	Jingpo	Spanish
Blackfoot	Kashubian	Sunda

Breton (France)	Kawa	Swazi
Bugotu	Kikuyu	Swedish
Catalan	Kiswahili (Kenya)	Tahitian
Cebuano	Kongo	Thai
Chamorro	Korean	Tok Pisin
Chinese (Simplified)	Kpelle	Tongan
Chinese (Traditional)	Kurdish	Tswana
Corsican (France)	Latin	Tun
Croatian	Latvian	Turkish
Crow	Lithuanian	Turkmen (Turkmenistan)
Czech	Luba	Uighur (Latin)
Dakota	Malagasy	Upper Sorbian (Germany)
Danish	Malay	Uzbek
Dutch	Malinke	Vietnamese
Dutch (Belgium)	Maltese (Malta)	Welsh (United Kingdom)
English	Maori (New Zealand)	Wolof (Senegal)
Eskimo (Latin)	Maya	Yiddish
Esperanto	Miao	Zapotec
Estonian	Minangkabau	**LANG + ENGLISH**
Faroese	Mohawk (Mohawk)	Chinese Simplified and English
Fijian	Moldavian	Chinese Traditional and English
Filipino (Philippines)	Nahuatl	Japanese and English
Finnish	Norwegian	Korean (Hangul)
French	Norwegian, Bokmål (Norway)	Korean and English
Frisian (Netherlands)	Norwegian, Nynorsk	**SPECIALIZED FONTS**

	(Norway)	
Friulian	Nyanja	MICR (E-13B)
Galician	Occidental	MICR (CMC-7)
Ganda	Occitan	OcrA
German	Ojibway	OcrB
German (Luxembourg)	Papiamento	**FORMULAS AND PROGRAMMING**
German (New Spelling)	Polish	Basic
Greek	Portuguese	C/C++
Guarani	Portuguese (Brazil)	COBOL
Hani	Quechua (Peru)	Digits
Hausa (Latin, Nigeria)	Rhaeto-Romanic	Fortran
Hawaiian	Romanian	Java
Hebrew	Romany	Pascal
Hungarian	Rundi	Simple chemical formulas

Please use the GetSupportedRecognitionCultures method of the Atalasoft.dotImage.Ocr.OcrEngine base class to obtain a full list of supported languages.

It should be noted, however, that the number of languages for which ICR is supported, is smaller than the total amount of languages supported. Attempting to use ICR on a language which does not support ICR, will result in an OcrException

A list of languages for which ICR is supported can be obtained via the GetSupportedICRRecognitionCultures. For convenience, a list of these languages is also provided below:

Albanian	Azeri (Latin)	Basque	Breton
Corsican	Croatian	Czech	Danish

Dutch	English	Estonian	Finnish
Flemish	French	Frisian	Galician
German	German (Luxembourg)	Greek	Hungarian
Indonesian	Irish	Italian	Lappish
Latvian	Lithuanian	Maori	Mohawk
Norwegian	Norwegian (Bokmal)	Norwegian (Nynorsk)	Polish
Portuguese	Portuguese (Brazilian)	Quechua	Serbian (Latin)
Slovak	Slovenian	Somali	Spanish
Swahili	Swedish	Tagalog	Turkish
Turkmen (Latin)	Uzbek (Latin)	Wolof	Xhosa

Output Formats

The AbbyyEngine supports the following list of output formats (provided here along with their corresponding MIME types):

Output document type	**Corresponding MIME type**
Plain Text (.txt)	`text/plain`
Rich Text (.rtf)	`text/richtext`

EPUB	application/epub+zip
FB2	application/x-fictionbook+xml
HTML	text/html
XML	text/xml
XML Paper Specification (.xps)	application/vnd.ms-xpsdocument
ALTO	text/xml-alto
PDF	application/pdf
Open Office word processing document (.odt)	application/vnd.oasis.opendocument.text
Microsoft Word 2007+ format (.docx)	application/vnd.openxmlformats-officedocument.wordprocessi
Microsoft Excel format (.xls)	application/excel
Microsoft Excel 2007+ format (.xlsx)	application/vnd.openxmlformats-officedocument.spreadsheetm
Microsoft PowerPoint 2007+ format (.pptx)	application/vnd.openxmlformats-officedocument.presentation

Deployment

The ABBYY FineReader engine requires all assemblies and support files within the resource archive that is distributed to customers of the add-on. The list of files is very large and a discussion on them is outside the scope of this document.

What should be noted, is that within the resources folder, lie the folders `Bin` and `Bin64`. These folders correspond to the resources for x86 and x64 processor architectures respectively. Which of these sets of resources will be loaded upon initialization, depends only upon which processor configuration of DotImage is installed/packaged with your application (x86 or x64).

Leave the folder and document structure unchanged; any changes could adversely affect the correct initialization of the engine.

It may be possible to reduce the overall file size by excluding language files and dictionaries for those languages that are not needed/wanted. Instructions for this process will be provided separately for customers of the add-on.

Reducing size of Abbyy OCR Resources

The full download of the Abbyy OCR Resources is approximately 1.1. GiB, and any application using AbbyyEngine will need to include the resources. Part of what makes AbbyyEngine so large is that it comes with a large number of language files. It is possible to significantly reduce the size of your resources. Please see the accompanying AbbyyEngine_README.rtf which shows the Data locations for the given language... the general pattern is that each language's dictionary files are under

AbbyyOcrResourcesFolder\Dta\ExtendedDictionaries\

Not every language has all three file types but they follow the pattern

LanguageName.amd

LanguageName.amm

LanguageName.amt

Limitations

No Stream Output

The AbbyyEngine supports nearly all of the standard functionality defined by the Atalasoft.dotImage.Ocr.OcrEngine base class; including recognition and translation – whether using foreign translators (TextTranslator and PdfTranslator), or the engine’s own built in translation functions.

The only core functionality that’s missing is the ability to translate to a stream output – the output of translation from the ABBYY FineReader engine must always be to a file.

No Auto-Rotate

INFO: AbbyyEngine - Overview

It is important to note that the AbbyyEngine, at least for its initial release – does not support the AutoRotate preprocessing option. Essentially, this means that the engine has no ability to correct nor recognize documents rotated at 90/180/270 degree angles automatically. Attempting to recognize or translate such rotated documents naively, will lead to incorrect and unexpected results.

There are ways to work around this limitation.

One such way is to perform rotation of the document via custom code, implemented within the ImageTransformation event handler of the Atalasoft.dotImage.Ocr.OcrEngine base class. In this case, responsibility for the identification of rotated pages as rotated by x angle, and for their subsequent transformation to an upright position – rests with the client. Note, that in this case, the co-ordinates of any text outputted by the Recognize methods, or the positions of text outputted by foreign translators in conjunction with the Translate methods – will be based on the dimensions and co-ordinates of the rotated image after its transformation, and not of the image as it was in its initial state.

It is not however strictly necessary to rotate the image. Recognition and Translation can be performed anyway, via one of two similar techniques

Recognition can employ the `Recognize(AtalaImage image, List<AbbyyTextRegion> abbyyRegionList)` method, which is specific to the ABBYY FineReader engine, and which is described above in the Features section.

This method essentially allows the specification of regions of text in images manually; including their orientation – which can be set during the construction of an AbbyyTextRegion subclass (AbbyyOcrTextRegion or AbbyyIcrTextRegion) via a corresponding OcrTextRotation value.

It is also possible, if it is known that the entire image/document is rotated, to simply pass in one AbbyyOcrTextRegion as a parameter with its bounds corresponding to the dimensions of the entire image, and with its orientation set to the corresponding rotation of the image. An example is shown below:

```
     new AbbyyOcrTextRegion(new Rectangle(0, 0, image.Width, image.Height),
OcrTextRotation.None)

};

var page = engine.Recognize(image, abbyyTextRegions);
```

If it is Translation that it is required, then it can be carried out via the PageLayout event; which is called after the engine has analyzed the image but before it has performed OCR on it. The client can register a custom listener to this event, in which the client can inform the engine directly about the locations and orientations of text on the pertaining image(s), using OcrTextRegion objects that are instantiated with orientations and bounds that correspond to the blocks of text they designate, or simply one OcrTextRegion whose bounds correspond to the dimensions of the entire image and whose orientation corresponds to the image’s rotation. This approach bears much similarity to the one described above for Recognition. Example code is shown below:

```
engine.PageLocation += (sender, e) =>

{

    var tr = engine.Factory.OcrTextRegion(new Rectangle(877, 282, 83, 132),
OcrTextRotation.Clockwise90);

    var tr2 = engine.Factory.OcrTextRegion(new Rectangle(400, 201, 80, 170),
OcrTextRotation.Clockwise90);

    e.RegionsOut = new OcrRegionCollection {tr, tr2};

};

engine.Translate(new FileSystemImageSource(“C:\\testimage2.jpg"}, true),
"application/pdf", "C:\\searchableRotated.pdf");
```

Parallel Processing / Thread Safety (prior to 10.7.0.10)

For the initial Release, AbbyyEngine has no support for Parallel Processing, and is not thread-safe. Starting in 10.7.0.10, our AbbyyEngine supports Parallel Processing. OU must set the AbbyyEngine.ParallelProcessing property to true

Even when using 10.7.0.10 or later and using the ParallelProcessing true setting, please ensure that all instantiations, initializations, deinitializations and object references relating the ABBYY FineReader engine – take place on one and the same thread.

INFO: AbbyyEngine - Overview

Setting the ParallelProcessing flag to true does not make the engine "thread safe" for you to enable multithreading directly, it tells the AbbyyEngine to use multiple threads internally. You should use the engine as before, but the internals will make use of multiple cores internally.

Multiple Recognition Cultures in Same Document (added 11.0.0.8)

Many customers requested the ability to support multiple languages on the same document - for instance, in Israel, its' common to have doucments containing both Hebrew and English. In the past, we had some special combination languages such as English and Chinese, English and Japanese, English and Korean.

Now, as of 11.0.0.8 you can specify your own combinations of languages and even have it recognize more than two

Instead of setting `engine.RecognitionCulture = someSingleCultureInfo;`

you will need to build a List<CultureInfo> with the desired languages and pass that in to the new engine.RecognitionCulturesList along these lines:

```
List<CultureInfo> supportedCultures =
engine.GetSupportedRecognitionCultures().ToList();

List<CultureInfo> targetCultures = new List<CultureInfo>();


//// EXAMPLE - mixed English and Hebrew:

//// this is how you can find and add supported languages to the list

targetCultures.Add(supportedCultures.Find(r => r.DisplayName == "English"));

targetCultures.Add(supportedCultures.Find(r => r.DisplayName == "Hebrew"));

//// instead of this single culture

//engine.RecognitionCulture = oneSingleCultureInfoHere;

// you now use this:

engine.RecognitionCulturesList = targetCultures;
```

INFO: AbbyyEngine - Overview

Original Article:

Q10432 - INFO: AbbyyEngine - Overview

Atalasoft Knowledge Base
https://www.atalasoft.com/kb2/KB/50068/INFO-AbbyyEngine-Overview