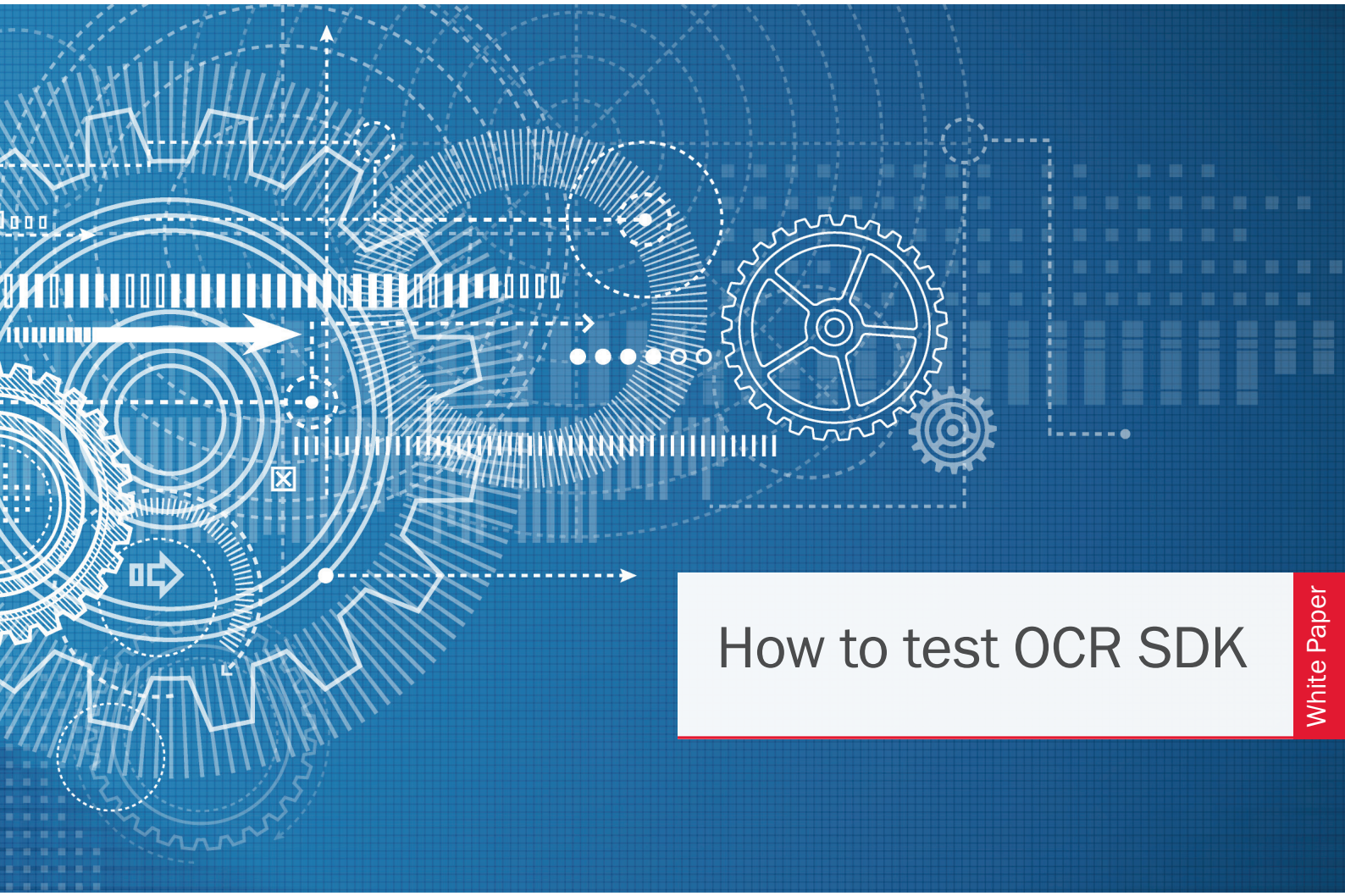
ABBYY®



How to test OCR SDK

White Paper

Contents

Introduction

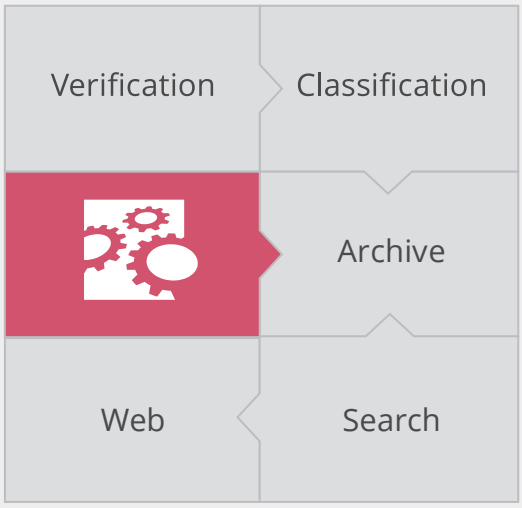
The choice of an SDK is a very important task because any decision made at the selection stage carries long-term consequences for your application and business. Replacing the technology on a later stage might be accompanied by difficulties. Therefore, a thorough testing and evaluation process is inevitable.

However, for following reasons, an evaluation of OCR SDK products might be challenging:

1. To test the OCR engine, you need a testing tool and a big database of sample images.
2. There are many OCR vendors offering SDKs, and, while there have been some public tests from reliable sources, most of these tests were more academic than practical as they were conducted under some general conditions. Relevant and practically applicable test results should be obtained in real-life conditions determined by the planned use case. We'll talk about this more below.
3. Several parameters must be tested, sometimes, in several languages: level of words/symbols accuracy, layout retention in MS Office formats, file size of created PDF, etc. Some of these parameters can be tested automatically while others can be checked only with the eyes. For different tasks/scenarios you might have to test different parameters.
4. To tune OCR for a particular task, a developer is expected in most cases to have at least a basic knowledge of OCR technology.

Years of working and interacting with developers testing OCR SDKs tell us how painful this process could be and that is the reason why we have decided to prepare this guide that describes key aspects of OCR SDK testing.

We are mindful of the fact that instructions produced by one of several OCR SDK vendors may raise some doubts. Therefore, we want to assure you that our goal for this guide was to be as objective as possible and include only the information that will be useful for the testing of any OCR engine. We hope that this guide will help you to test OCR SDKs carefully and choose the best one for you.



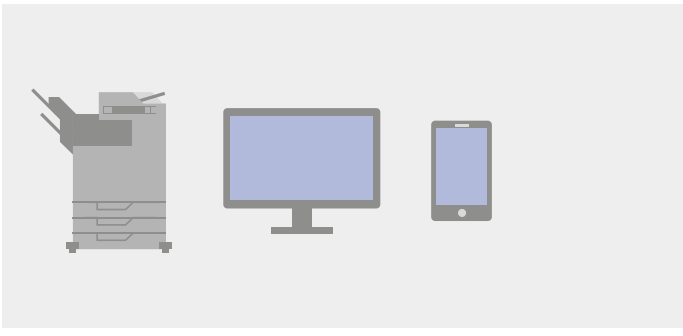
Before you start

There is no clear answer to the question of how accurate a particular OCR SDK is since the key characteristics of an OCR SDK cannot be measured in vacuum. Just as speed and efficiency of a car sometimes depend on road conditions or fuel quality, the accuracy and speed of an OCR SDK are quite dependent on the task and on the set of technical conditions such as servers' parameters, operating system, scenario, document types, document quality, just to name a few.

Image base

At first, you need to prepare an image base for testing. The key things to remember are:

1. **Large image base.** Your image base should be large enough. For highly reliable results, we recommend you to collect several thousands of images. If this is too complicated, collect at least several hundreds as the number of images influences the reliability of the test result. If your test base only contains, for example, 100 document pages, a single document page already represents 1% of accuracy and has a strong influence on the overall accuracy measurement result (if you measure in documents, see "How to calculate accuracy").
2. **Document type.** Collect the images that correspond to the document types you'll process in production. If, for instance, you are going to process invoices, collect invoice document samples, and, if you plan to process agreements, then collect agreement document samples, etc. This is important because different OCR engines tend to work better with different types of documents (because they were originally created to work in different scenarios). We recommend to use the same proportion of document types in the testing as will be in production. This will bring you near the accuracy rate that can be later expected for your task in production.
3. **Image source** (scanned documents, mobile photos, different types of PDFs). Test only on images that were captured the same way they will be captured in production. For instance, it will be of no use if you test the OCR SDK on scanned documents while you plan to process photos captured by smartphones during the production stage. The reason for this is the same: different OCR engines work better with documents from different sources.



4. **Real documents.** The best way to test OCR engine is a test on real samples of documents you are going to process in production. Testing an OCR engine using fake documents artificially created just for this testing will not lead to realistic results. Also avoid using images provided as a test base for one OCR SDK for testing of another OCR SDK. Each vendor will provide you with the images that are OCRed well by his product, or even with the documents that were used for tuning of his SDK. The results may be worse with products of other vendors.

Accuracy measurement

How to compare OCR results with the original text

There are several methods for comparing OCR results with the original text:

1. **Eyeball estimation.** Review the OCR result and rate it somehow. Advantages: this method is fast and lets you deselect the most unsuitable engines from your list. Limitations: such testing can't be done on hundreds of images, as it's too time consuming. So, you won't get statistically reliable result.
2. **Track changes in Microsoft® Word.** Advantages: this method is suitable for testing on small number of documents. Limitations: low testing volumes and unreliable result. In addition to your source documents, you need to have original documents in DOCX file format in order to compare them with the OCR results.
3. **Peek-a-Boo.** Advantages: this method is suitable if you are going to convert images into editable formats, and if the OCR engine you are going to test provides good layout retention. Limitations: only suitable for low testing volumes, unreliable result. We strongly recommend that you don't use this method for data capture.
4. **Verified base of source images** + parsed results converted into XML format (reference data). Advantages: this method will give you the most statistically reliable results. It's language-independent and suitable for high-volume testing. Limitations: this method requires some additional time for test base preparation.

In this document, we describe accuracy measurement using the fourth approach since it gives the most reliable and repeatable results.

What to measure

There are several ways to measure the accuracy:

1. By measuring the percentage of **symbols (characters)** recognized correctly.
2. By measuring the percentage of **words** recognized correctly. The word is deemed to be correctly recognized only if all the symbols in the word were correctly recognized.

NOTE: For the approaches 1 and 2 you will need a test base containing reference data with correct symbols and their coordinates.

3. By measuring the correct detection rate of the structure of the document.

NOTE: For the last approach, reference data in your test base should contain each document structure in addition to correct symbols and their coordinates. This can be defined in XML via special entities specified for header, footer, text column, print, image object, background, etc.

Character error rate can then be computed as:

$$CER = \frac{S+D+I}{N} = \frac{S+D+I}{S+D+C}$$

Where:

- S is the number of substitutions,
- D is the number of deletions,
- I is the number of insertions,
- C is the number of the corrects,
- N is the number of characters in the reference (N=S+D+C).

Word error rate can then be computed as:

$$WER = \frac{S+D+I}{N} = \frac{S+D+I}{S+D+C}$$

Where:

- S is the number of substitutions,
- D is the number of deletions,
- I is the number of insertions,
- C is the number of the corrects,
- N is the number of words in the reference (N=S+D+C).

See more here (https://en.wikipedia.org/wiki/Word_error_rate).

To calculate S, D, I and C you need to calculate **Levenshtein distance**, for example, with a help of the following algorithm:

```
// len_s and len_t are the number of characters in string s and t respectively
int LevenshteinDistance (const char *s, int len_s, const char *t, int len_t)
{
  int cost;

  /* base case: empty strings */
  if (len_s == 0) return len_t;
  if (len_t == 0) return len_s;

  /* test if last characters of the strings match */
  if (s[len_s - 1] == t[len_t - 1])
     cost = 0;
  else
     cost = 1;

  /* return minimum of delete char from s, delete char from t, and delete char from both */
  return minimum(LevenshteinDistance(s, len_s - 1, t, len_t    ) + 1, /*insertions*/
                 LevenshteinDistance(s, len_s    , t, len_t - 1) + 1, /*deletions*/
                 LevenshteinDistance(s, len_s - 1, t, len_t - 1) + cost); /*substitutions*/
}
```

See more here (https://en.wikipedia.org/wiki/Levenshtein_distance#Computing_Levenshtein_distance). It is recommended to apply this algorithm only if the order of the words is clear. For example, apply it to the blocks of text that were detected on the page, but not to the whole page.

Choosing Optimal Approach. Your choice of a particular method/approach depends on your project, or scenario.

Searchable PDF. If you are going to convert images into searchable PDF, then the percentage of **words** recognized correctly should be measured, because end users will use the OCR results for searching for whole word, not symbols.

Editable formats. If you are going to convert images into DOCX, XLSX, etc., then you'll need to measure the percentage of correctly recognized **symbols** and evaluate the level of layout retention.

Data capture. . If you are going to implement your own data capture based on OCR results, it's better to measure percentage of correctly recognized **words**.

Or it might be even better to separately measure percentage of correctly recognized **fields** and the average number of errors in fields with errors. This will give the idea of how many fields will require manual editing and how much editing per field will be required.

For data capture, a good practice is to define data of paramount importance on documents. For example, there is no particular need to recognize footer on an invoice, but it's critically important to find the fields like "Invoice date" and "Total". Keeping this in mind will enable you identify which OCR engine will work better for your particular task.

Data capture normally requires a higher level of accuracy than other scenarios. In a data capture scenario, reference data should include key words that are used for locating fields and values of such fields.

How to calculate accuracy

There are several ways to calculate average accuracy for a set of documents:

1. Determining the percentage of correctly recognized symbols/words found in the whole testing document set.
2. By first calculating how many correct symbols/words were found in each document and then calculating average percentage for all documents in a set.
3. For multipage documents: first measure accuracy for each page separately, then average accuracy for all pages of the document and finally average accuracy for all documents in a testing set.

There is one more aspect that is worth mentioning – **mistake penalizing logic**. In general, each incorrectly recognized symbol/word would mean minus 1 point. However, some types of errors are less critical and can be ignored, while others might have severe consequences in the planned processing scenario and should be penalized by more than 1 point.

Choosing a suitable method: the choice of calculation method and penalizing logic depends on the intended scenario.

Searchable PDF. If your scenario is conversion into searchable PDF, it's better to first calculate the percentage of correctly recognized symbols/words found in each document, then calculate the average percentage for all documents in a set. It's important to understand how many documents containing particular key word will users be able to find. Penalizing: punctuation should be ignored in the penalizing logic, because it is not used for search.

Data capture. For this scenario, it's better to calculate percentage of correctly recognized key words and correctly extracted field values that were found in each document, then to calculate the average percentage for all documents in a set. These numbers allow you to understand how many documents could be captured automatically with 100% accuracy and how many fields would require verification.

For errors found in the values of key fields on the document, the result should be penalized more strictly. On the other hand, errors in the key words that are used for locating the fields aren't so critical if these key words can at least be found correctly and if the errors are repeated more or less consistently from one document to the other. Such key words can later be re-OCRed using two-pass OCR method.

Editable formats. For this scenario it's recommended to calculate how many words, tables, footers, etc. were recognized correctly for all documents in a set. There is no use in calculating these metrics for every document because the only important thing in this scenario is to estimate how many corrections the application's user will have to do in total.

Speed measurement

It's important to bear in mind that OCR process consists of the following steps:

1. Engine initialization
2. Image processing (includes preprocessing, analysis, recognition and synthesis)
3. Engine deinitialization.

Each of these steps may significantly influence the final speed criteria. However, not all of these steps are applicable to every scenario.

For example, when processing large set of documents in one go (batch processing), there is no need to initialize the engine for each document, therefore there is no need to measure initializing time for each document processing. If in production the images will be transferred for processing via RAM (to ensure high speed) then the same logic should be implemented in your testing algorithm. There is no use for testing in conditions when images are opened from disk in that kind of situation, as the final speed criteria will differ significantly.

That's why it is important to only measure speed for those steps that you anticipate in your production.

Several tips:

1. For getting more reliable result, it's recommendable to run speed test several times and calculate the average time, because results may vary by multiple percentage points depending on interaction with the operating system.
2. Always use the real samples of the documents that you are going to process in production. Test under real-life settings. For example, if you plan to process BMP files, don't use JPG files in a test set as the speed may differ a lot.
3. Be sure you study the API, code samples and tutorials (if any) of an OCR SDK you are testing to understand what tools it provides for speed optimization (e.g. for objects reuse). Never use the engine's default settings, if speed is critical for you but optimize it according to your needs.

4. Try out various settings to see what can be switched off to increase the speed.

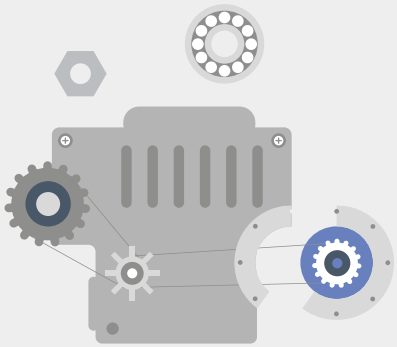
Product distribution size

The SDK distributive size is not always at the top of priorities, but sometimes, it is an important parameter:

- **Ad-hoc scenario.** The distributive size influences initialization time, so if the engine is started every time a new task comes in for processing, we recommend that you check the distribution size.
- **Mobile applications.** Distribution size of mobile applications will always be an important consideration for mobile application developers, which is why OCR engine size is one of the most critical parameters.
- **Browser plugin.** The bigger the distribution size, the larger the plugin size which can lead to longer installation time for browser plugins.
- **Drivers.** Since drivers are usually updated regularly via Internet, so larger distribution size may call for additional charges. Driver may sometimes be kept running in the RAM, which has a limited size. If a driver is uploaded on request into the RAM, a large distribution size will lead to additional delays.
- **Web-service.** If OCR is integrated into a web-service, the SDK distributive size will negatively impact the speed of scaling the service up and, consequently, influence the amount of processing power you will need to provide adequate scalability.

There are 2 main ways to decrease an SDK distributive size:

1. Exclude dictionaries for the languages that aren't so important for your task.
2. Exclude parts of engine's code that improve accuracy but aren't so critical, so that after the exclusion, the level of accuracy will still be acceptable. You should decide for yourself which level of accuracy is acceptable for your task, depending on your scenario and users expectations. Unfortunately, making exclusions is not always possible or easy. In many cases a custom SDK distributive will have to be created.



Vendor's support and reliability

Technical characteristics are not the only important things to consider when choosing an OCR engine.

Is the OCR engine well supported? Find out how your possible vendors handle feature requests, check availability of good documentation and code samples and the level of knowledge of their engine by support personnel.

Also find out if the company has ever done projects similar to yours (ask for references, check case studies and the web-site). This could indicate the level of expertise the vendor possesses in your particular scenario and in your document types. OCR support is not a trivial task, that's why expertise in a particular field matters.

Is the engine updated regularly? Some OCR engines have not been upgraded for years while others receive regular updates. Updates are important because they include bug fixes and new functionality requested by customers or required to conform to industry changes. If you plan to use OCR in your business be aware that, at some point, you'll want something to be fixed or improved in the engine.

We hope that the above recommendations will help you to make the right decision based on clear and repeatable facts. If you have any additional questions or want to provide your feedback, please contact us at http://blog.ocrsdk.com/how-to-test-ocr-sdk/!

ABBYY

ABBYY International Headquarters
Otradnaya str. 2b/6 127273, Moscow, Russia
Phone: +7 495 783 3700
Fax: +7 495 783 2663
E-mail: office@abbyy.com
Support: support@abbyy.com