Improving OCR and ICR Accuracy Through Expert Voting

A White Paper

Presented by
Michael Breithaupt
Océ Document Technologies

791 Piedmont-Wekiwa Road
Apopka, FL 32703
Tel: (407) 598-1800
Fax: (407) 598-1879
Email: info@csisoft.com
Web: http://www.csisoft.com

Background

OCR (Optical Character Recognition) analyzes the shape of a bitmapped character and assigns a value to it. Different OCR products use differing methodologies usually based on a system of template matching or a mathematical analysis (consisting of feature analysis and feature extraction) as their baseline methodology. These analyses usually produce a range of possible results, so they are supplemented by post analysis validations which support the most likely result followed by the possible alternatives. Each possible character is supported by a likelihood percentage. It's an iterative process within an engine with analysis and check performed multiple times – sometimes as many as 10 within one engine to derive the most likely result.

Simple Fonts give High Accuracy

Most of today's OCR software can produce highly accurate results with well-formed laser or good quality machine printed text. This is particularly true with simple fonted 10-16 point fixed text and fixed character spacing on a plain background. However variable widths, proportional fonts, kerning, large sized text or exotic fonts will reduce accuracy and handprinted characters pose even larger challenges.

Differences between Full Text OCR and Forms Processing

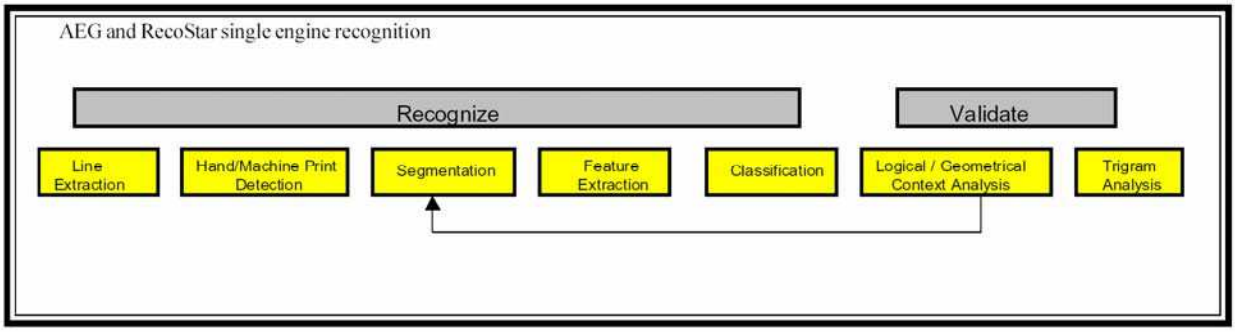
Full text OCR is designed to convert a page of similar machine printed textual elements interrupted by photographs or diagrams, often formatted into two or more columns. The software needs to understand and decode this formatting as well as identify and capture the fonts used so as to enable easy editing. Legal conversion systems which need to understand the formatting of a case would be an example of this. Forms processing OCR is designed purely to capture transactional data from a form in an ASCII format typically to update a back-end computer system.

Forms Processing poses Challenges

In forms processing the challenges are greater. Data on forms can be created from carbon or carbonless forms, the printer may be a dot matrix, the original scanned forms may have been a fax, the background of the form may interfere with the foreground. Fields may not have dictionary entries to look up. Some fields may be created with either constrained handprint or worse, with unconstrained handprint. Sometimes a field may have mixed data types and frequently the field may contain either machine print or handprint which varies from form to form.

Accuracy Statistics and the Problem of Substitutions

While everyone wants accurate conversion, accuracy is a difficult concept. Some people define the accuracy rate as the percentage of all characters output as "recognized" by an OCR engine regardless whether the character has been correctly recognized or not. In CSI IntelliDact's world, the accuracy rate contains all the characters output by an OCR engine which have been recognized *correctly*. We know the correct characters, because we analyze this against a pre-known 'truth file'. This is a different approach from that of many other vendors who simply provide a percentage of those characters which the engine thinks are right. Thus with IntelliDact, the characters which have been output AND have been wrongly recognized are defined as errors or substitutions. As the engine also outputs characters which are flagged as low-confidence characters, a third category is the 'defined rejection rate'. Together, the three categories rates add up to 100%. One of the parameters available from a good OCR engine is the acceptance threshold, which allows the user to manipulate the substitution over the rejection rates. Generally speaking, a low acceptance threshold returns more "recognized" characters and contains more errors while a high acceptance threshold will do the opposite. So if the acceptance threshold is set too low, the engine will accept a very high percentage of characters and may include some characters which it thinks are correct, but which are wrong. These are known as substitutions and represent the most expensive errors to correct. On the other side, setting the acceptance threshold too high results in more rejected characters. Even though most of these rejected characters may have been recognized correctly, they need to be verified in a very labor-intensive post-processing step. Eliminating substitutions at a low rejection rate should be the true goal of a good OCR engine.



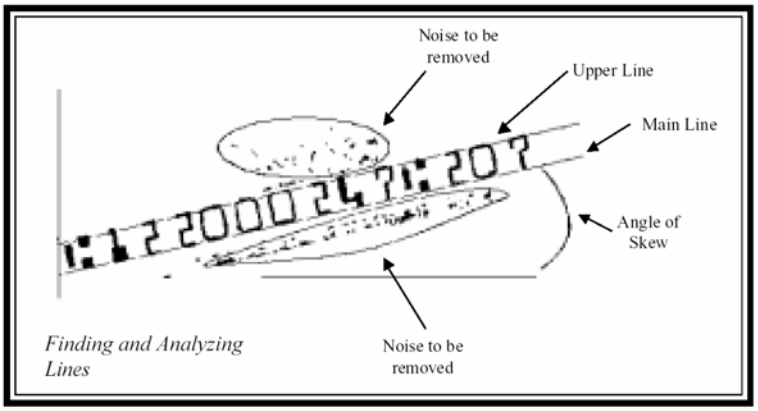
Voting Eliminates Errors and Improves Accuracy Rates

Voting takes the output from two or more recognition engines and compares the results, voting on the most likely. Voting is often cited as a method to try to improve the recognition accuracy from difficult types of images, however this is inaccurate. Voting is designed to eliminate errors and/or increase accuracy percentages at the same time. The preference in an OCR application of whether to get less errors at the same *accuracy* level, or higher accuracy at the same *error* level is controlled by various switches within the OCR engine. All OCR engines produce more than one result – and assign likelihood percentages to each result. Voting takes the recognition results from multiple engines and compares them – in some cases eliminating an engine, in others combining them to improve the results. In forms processing applications and in handprint applications, voting can be remarkably attractive.

The Basic Building Blocks of OCR

Recognition

The OCR recognition process consists of two major steps: extract and recognize the characters and then prove that the result is within its given context. If it does not pass its validation, then try the next best alternative until there is a high likelihood of accurate conversion. Within the recognition phase there are four main steps: Line Find, Character Segmentation, Feature Extraction, and Classification.



Line Extraction

In the case of forms processing, the identification of multi-line field blocks such as name and address or tables is critical. For each block, the first step is to group all data elements (areas of connecting pixels) within the context of baselines (see illustration) so that the following steps are sure that they are dealing with complete characters. In the case of IntelliDact this is achieved through finding imaginary baselines and the rotation angle of the line. Although this may be difficult with handprint due to its up-and-down nature, once identified the baselines can be used to remove data elements (noise) pixels that fall outside them, improving segmentation and recognition.

Hand/Machine Print Detection

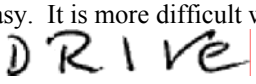
The varying distance of characters from the baselines and the varying character heights tells the software whether the characters are machine print or handprint, since handprint tends to go up and down whereas machine print tends to be level. If the algorithm cannot unequivocally decide whether a field is hand or machine print IntelliDact recognizes the field through the machine and hand print classifier and then decide at the very end which result will be output.

Character Segmentation

Segmentation is the process of separating the characters. There are completely different segmentation algorithms for machine print and for hand print, and machine print algorithms vary for variable vs. fixed fonts. The first step is to determine the type of segmentation to perform. Is it:

- Fixed pitch characters? Such as in `HELLO` where each character has the same spacing
- Variable pitch? Such as in HEIGHT where the spacing varies depending on the characters
- Don't know?

If the algorithm cannot determine as to whether the field has a fixed or a variable font, IntelliDact performs the segmentation for both type of fonts, and decides at the very end which result will be used for further processing.

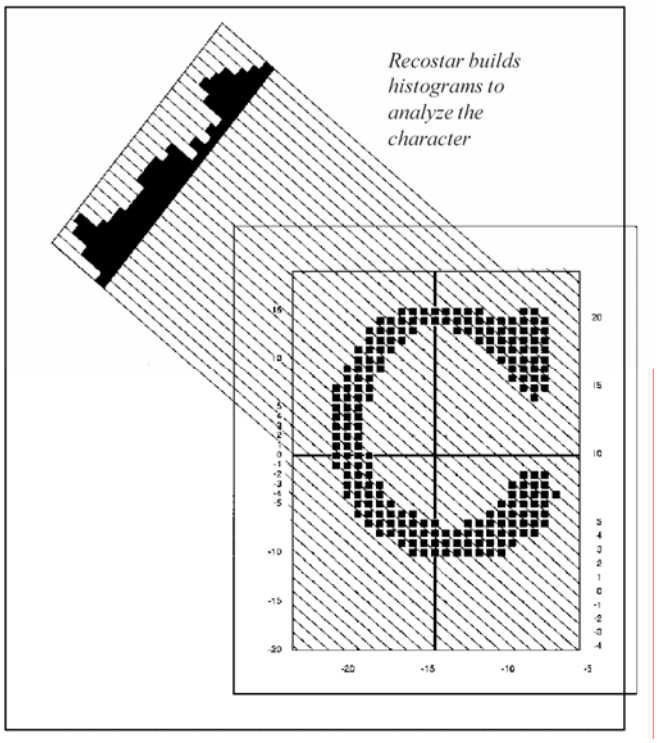
In the case of fixed pitch segmentation this is easy. It is more difficult with variable pitch fonts and most difficult with handprint. For example consider the word  as scanned here. It looks fairly straight forward. But the letter 'v' overlaps with the letter 'e'. All character elements are stored as run-length-coded (RLC) objects, which means each isolated data element can be moved, removed or logically connected with another RLC object to form a character. Initially the IntelliDact looks at a vertical gap between RLC objects or follow any "white path" between RLC objects to determine all RLC objects belonging to a character. Therefore in the word DRIVE above it might find 7 RLC objects –two from the D (circle and vertical bar), two from the R (vertical bar separated from the rest of the character) and one from I, V and E respectively. It then uses a histogram distribution of the

pixels to make the determination of where the most likely break in characters is or which RLC object belongs to which character. This is then subsequently classified, with logical and geometric context (as defined later) used to determine the right break or cut of the segmentation process. For example as v and e overlap the segmentation process would propose two alternative segmentations: l) v and e as one character, 2) v and e as separate characters. Both proposals will be classified and validated. In our example, the result of the combined character v and e will have a low confidence character as a result, while the separated v and e will have good results with high confidence levels.

Feature Extraction

Feature extraction is the first step in classifying a character shape. AEG and RecoStar use fundamentally different approaches to analyzing the features of a character shape to perform this task. The difference in methodology is the foundation of a successful voting system as it compensates for the weakness of each engine while it combines the strength of all the engines involved.

The principal ideas of feature extraction are: (a) all features have to be complete; two different characters have to be clearly differentiated solely based on the features describing them; and (b) features have to be steady. For example, if a character shape is insignificantly distorted by some noise, the features describing the character should also just change negligibly.

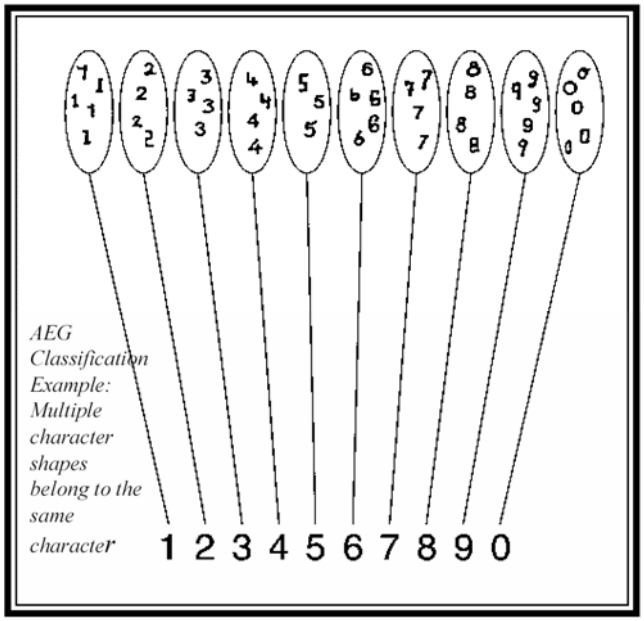


In the AEG product the scanned character is normalized within a 16xl6 matrix array, set to a common width and finally an artificial 16 bit depth (over 16,000 shades) grayscale is applied to each character to better analyze the shape.

The RecoStar product works somewhat differently, slicing the topography of the character every 15 degrees (giving 12 cuts) and looking at the shape to build a series of histograms determining the numbers of intersecting lines (similar to tomography in the medical arena). The data is used then for further mathematical analysis – a method also known as "Winkelschnittanalyse" WSA.

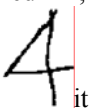
Classification

To unambiguously recognize a shape of character two requirements are fundamental: (a) the ability to differentiate characters of different shapes and (b) similar character shapes need to be classified as the same class of characters.



For these reasons IntelliDact recognition engines train on 10,000 to 30,000 different possible shapes for each character, e.g. 10 digits (one digit is one character class) can be broken down into more than 25 different classes of shapes. For example take the number 4. Let's consider 4 different possible shapes for the number 4:

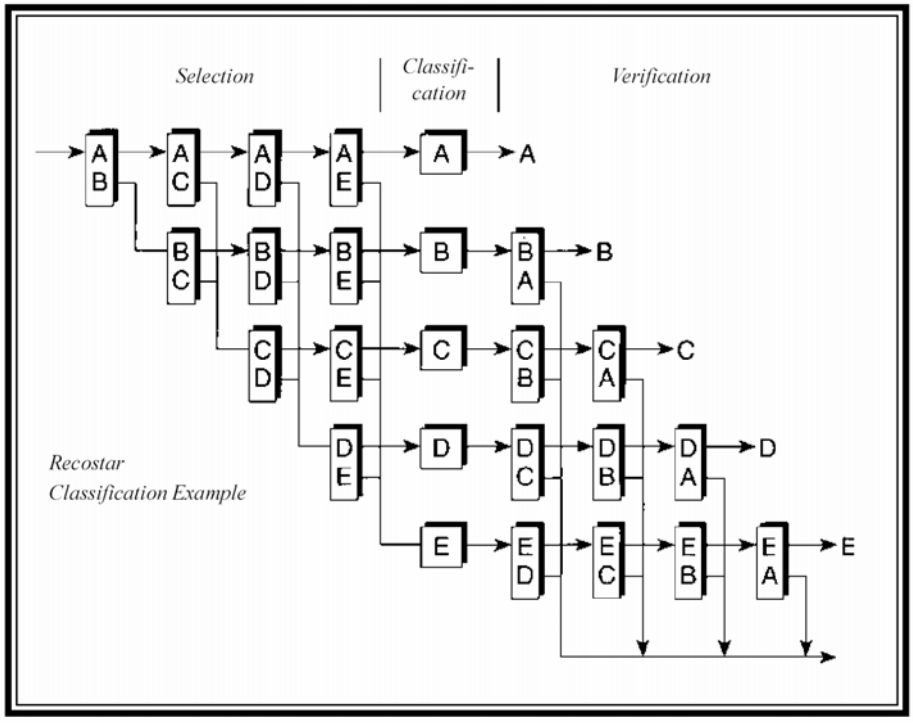
These may be labeled 4-1; 4-2; 4-3 and 4-4. When



the system finds a it might decide that this has a 95% chance of being type 4-1 as it shares the most commonality with this class of shape. The more classes of shapes are defined in a classifier the more robust the recognition of different hand writing styles due to regional, ethnic and age influences.

RecoStar has trained a series of base line classifiers which contain 2 shapes to cover all eventualities. For example AB, AC, AD etc., then BC, BD etc. If for example you send an

to the first classifier which determines that it is NOT a B, then B is eliminated with the next check being an AC. This provides a high level of confidence in finding the correct character class and possible alternatives.



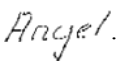
Validation or Proving the Accuracy

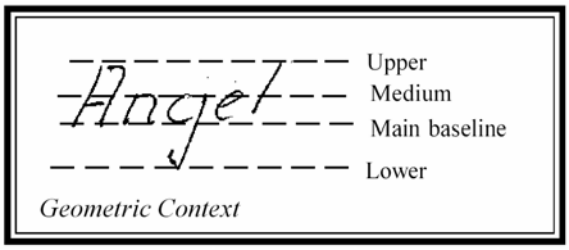
The context of the field helps to narrow down multiple choices of the OCR engine, i.e. it eliminates the ambiguity of certain characters (e.g. zero vs. "O"). Context can be defined by User (formal context) or can be automatically implied through an expert system (geometric and logical context) which analyzes the data recognized. All three context analysis tools work together hand in hand and comprise hundreds of rules packed into an expert system.

Logical Context

"Logical context" usually applies only in alphanumeric fields. The main task of logical context is to determine whether a group of characters (a group is defined by all characters between spaces or other delimiters like commas and dots) is a word (alphabetic only like MONUMENT), a number (digits only like 12500) or a "mixed" word (alphas and digits like WIN98). Consider the "O" in MONUMENT. Once "logical context" concludes this group of characters must be a "word" the character "O" loses its ambiguity and the also possible recognition result "0" (zero) will be eliminated. Logical context is also used to determine whether a character is being recognized as lower or upper case. This only applies to characters whose lower and upper case have the same shape (S vs. s, O vs. o, C vs. c) or the shape of a lower case character may be confused with the shape of an upper case character (I vs. 1). This kind of ambiguity may be resolved by applying general spelling rules. "Logical context" also uses "geometric context" to obtain further conclusions.

Geometric Context

To understand "geometric context" consider the word  . As an alphanumeric field the character  in this field may get a 55% confidence result of being a G and a 45% confidence factor of being a 9. In the case of IntelliDact, geometric context looks at the upper, medium, main base and lower baselines of a word as shown at left to determine whether each is an Alphabetic or Numeric character. In this case, if it was a 9, then it would fall between the upper and main baselines, but a G would fall between the medium and lower baselines. In this case logical context, in conjunction with geometric context, concludes that the field is alphabetic, the 9 disappears from the result list and the confidence level for G will be elevated. Geometric context can also give the engine a clue as to whether the character is upper or lower case. For recognizing amounts geometric context is heavily used to determine "1" and "," as the hand printed shapes of these characters very often look the same.



Formal Context

"Formal Context" checks on the result based on user-defined edit patterns such as NN-NN-NN for a date field.

Trigram Analysis

In the case of alphabetic information, three adjoining characters are analyzed in a process known as "Trigram Analysis", which utilizes a language dependent set of tables to decide on the acceptability of the three characters. So for example, the letters MZD cannot appear in an English language word, so if the primary selection produces these characters the engine will try the next most likely combination.

Dictionary Lookup

The completed fields are compared to defined dictionaries which can be individually set up for each field with the entire phrase; a partial match; or just the alphabetic portion of a field which is useful for street addresses. But not all spellings of a word may have been included so partial matches to the dictionary will adjust the 'confidence' factor accordingly.

The following handprinted text:  spells the word HELLO the German way. The dictionary will not find the entry, but in the AEG and CGK solutions it will find the closest entry and know that there is one character incorrect and adjust the confidence accordingly.

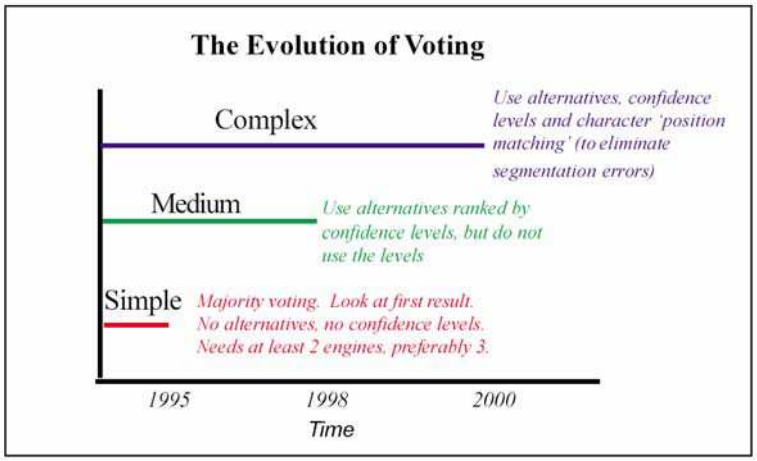
What influences the results of a single engine?

The difficulty of Character Recognition can be classified based on the following criteria:

1. **Handprint** is more difficult to recognize than machine print
2. **In machine print**, dot matrix or computer line printer produced characters are more difficult than laser printed or typewritten. Carbon varies but NCR paper can be faded and very difficult
3. **A1phabetic Characters** are more difficult than numeric and alpha/numeric is more difficult than alphabetic.
4. **Unconstrained** or variable pitch fonts are more difficult than fixed fonts
5. **Lower case** is more difficult than upper case

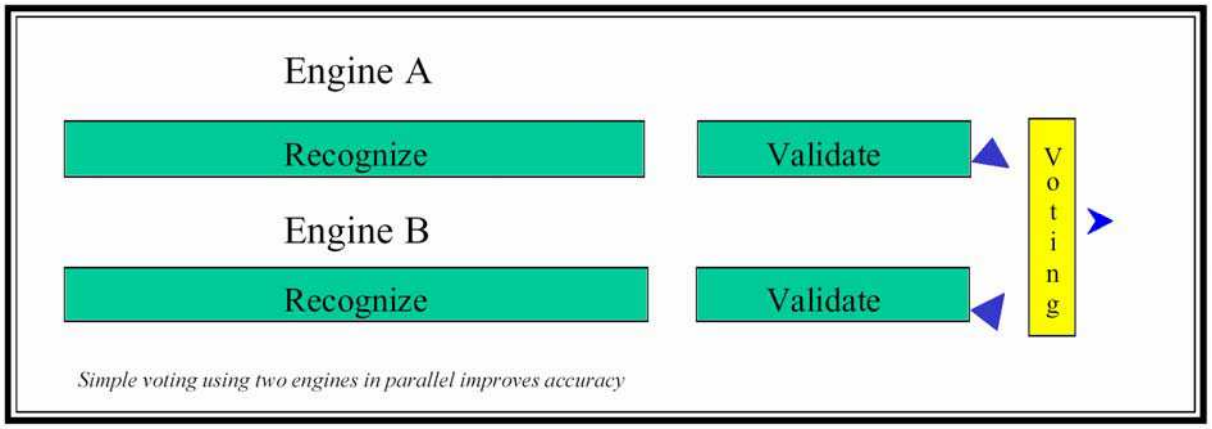
Voting for machine print and handprint

Voting, using the results of more than one OCR engine, can substantially help results on the harder and hardest types of characters to recognize. Today's processor power allows the software to be run many times – in fact in many cases the IntelliDact engines have gone through 10 iterations in order to interpret just one character. It has been shown as preferable in the case of fixed and variable fonts, to run both processes and determine the field confidence from looking at both results.



How voting works

Voting leverages from using the answers from more than one OCR engine to increase accuracy. It has evolved over the last few years from simply using two or three separate engines with majority voting to leveraging from an understanding of the internal processes of each engine. To appreciate this it is useful to review the different voting techniques in use today.



Simple Voting

A simple voting algorithm will determine that ***H*** is the character based on the majority ranking alone and not on the confidence factors. It needs at least two engines to work, but three engines produce better results. Depending on how many engines the system runs, the likelihood can be adjusted accordingly. It is a simple and effective way for manufacturers of forms processing to reduce errors, but it is possible to further improve performance by leveraging from the confidence.

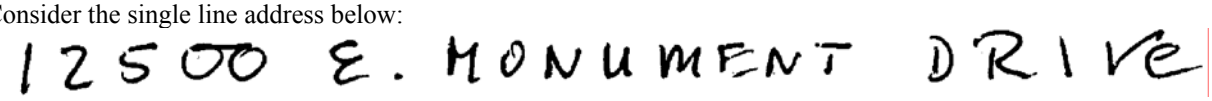
Use of Confidence Levels

The next level of voting leverages from the confidence levels reported by the OCR engines. In this case you do not need more than two engines, as the system has a lot more information to work from. However, to make confidence levels work, the vendor of the voting system must first identify commonality by normalizing the confidence levels of the various engines from each manufacturer.

The best way to do this is to run a huge test deck of predefined characters, comparing the results and storing 'credibility' adjustment factors. The normalization process can then classify the engines in a fairly simple way on the basis of the numbers of substitutions and the rejections found as shown in the graph.

This has been shown to decrease inaccurate conversions, but it is essentially static. Because the vendor has no internal control over the internal processes of the defined engines, he is unable to run a process modifying and optimizing the results.

Consider the single line address below:



- Engine 1 – the RecoStar engine – interpreted this as 1251#0 E. HONUMENT DRIVE
- Engine 2 – the AEG engine –interpreted it as 12500 #. MONUMEN-, DRIV##
- The voting engine –RecoStar Pro interpreted it ***correctly*** as 12500 E. MONUMENT DRIVE

There are three typical OCR problems identified in this example.

- **Bad Segmentation** – as when the RecoStar engine wrongly segmented the two zeroes and when the AEG engine wrongly segmented the V and E in DRIVE.
- **Poor Recognition** – which is shown by the # symbol, which indicates too low a confidence level or the confidence level of the first and second choice are too close to make a decision.
- **Substitution** – as when the RecoStar engine converted the M in MONUMENT to an H.

As an address line we were unable to tell the classifier whether the field was alphabetic or numeric, so each engine was looking for both character types. Clearly both engines individually had difficulty with the numbers as well as the T at the end of MONUMENT. The AEG engine also had trouble with the overlapping V and E at the end of DRIVE. But the voting engine eliminated the problems.

To understand why, we must look more closely at each engine's interpretation. Both engines can classify their confidence between 0 for lowest and 255 for highest.

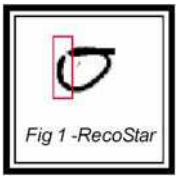


Fig 1 -RecoStar

Starting with the number  . The 00 is joined, but if you look closely you will see there is a break at the top of the first 0. The segmentation engine on the RecoStar engine has segmented the first part of this 0 into a 1 as shown in Fig. 1 and it had a very high confidence that it was right (255). It then had great difficulty classifying the remainder of the zero with the best guess being a 3 with very low confidence (1) see Fig. 2. It then identified the second 0 correctly (Fig. 3) with a very high (255) level confidence.


Fig 2 - RecoStar

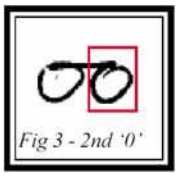


Fig 3 - 2nd '0'

The AEG segmenter worked differently. It correctly segmented the first 0 (see Fig. 4) and correctly interpreted it with 119 confidence that it was right. It was then left with the second 0, which it was not sure of, offering three alternatives – a 0 with a reasonable confidence of 145, an 8 with low confidence of 8 or a 5 with a low confidence of 7.


Fig 4 - AEG

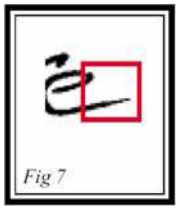
Note that the confidence factors are not percentages, they just represent the confidence that the engine has in each particular choice.

The M in Monument was a substitution wrongly interpreted by the RecoStar engine as an H (low confidence 65) with an M alternative as confidence 44. The AEG engine had a 185 confidence that it was an M with no alternatives.

Then we get to the E, which was correctly interpreted by the RecoStar engine with a very high confidence of 255. But the AEG result was not clear. It came up with an E with confidence of 31, an 8 with a confidence of 31 and an F with a confidence of22.

The  also produced a different result with the AEG engine separating out the top of the T from the bottom, coming up with a dash (213 confidence) and a very confident comma (255).

Lastly the V and E in DRIVE caused differences in result. The RecoStar engine confidently and correctly decided that the characters were V and E (confidence 255 in each case). But AEG was not so sure. It cut the top of the V (see Fig. 5) and produced a 213 confidence that its choice was correct. This left it with a problem as shown in Fig. 6. It decided that this strange character might be a B – confidence 41, or an E – confidence 26, or possibly a Z – confidence 25.

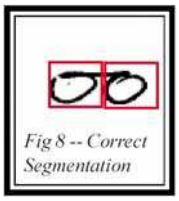


It then was left with yet another small set of pixels (see Fig. 7) which it tried to resolve. But it was not very happy with any of the alternatives. It decided that this might be an I (confidence 38), an S (confidence 34) or an L (confidence 29).

So how did the RecoStar Professional voting engine resolve this into the correct interpretation of:

12500 E. MONUMENT DRIVE

First the voting engine corrected the segmentation (see Fig. 8). The first zero in the house number then got resolved as a 0 (zero) with a confidence of 225. Although the RecoStar engine had a high confidence of the first part being a 1, this got outvoted because the confidence level of character following the 1 is very low and does not match the geometric position of the high confidence character as recognized by the AEG engine. In this case the results of RecoStar are not considered at all.

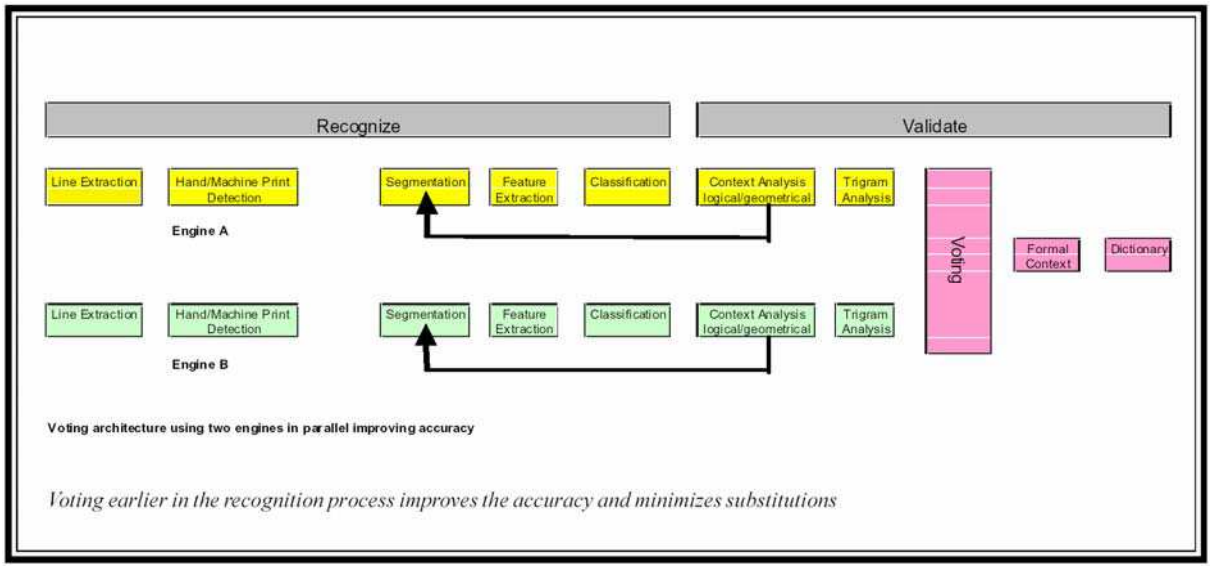


The second 0 got voted as a zero (confidence 248) or conceivably an 8 (confidence 5) or 5 (confidence 4).

Second, the E got resolved fairly simply as the RecoStar engine was confident and the AEG had it in its choice, albeit not first.

The substitution of the H instead of the M was resolved by the voting engine as an M (confidence 118) or an H (confidence 34).

In the case of the T, RecoStar Professional uses the internal location coordinates of the characters. So although it thought that the shape of characters might conform to a *dash* and a *comma*, this seems unlikely when voting on the result, as the top of the T was on the upper baseline with the lower part between the medium and the main baseline (see Page 6). A dash would typically be between the medium baseline and the main baseline, while a comma would typically fall between the main baseline and the lower baseline. As the surrounding boxes of both the dash and the comma of the AEG engine match the geometric location of RecoStar's T, a segmentation problem had been indicated to the voting engine and so it chose the T result over the less plausible dash/comma. One could argue that the segmentation problem should not have happened in the first place, but it also demonstrates that the voting algorithm is capable of ruling out certain incapabilities in either of the two engines. Likewise in the case of the V and E, the segmentation problem got solved through comparing the confidence levels and the matching positions of the characters.

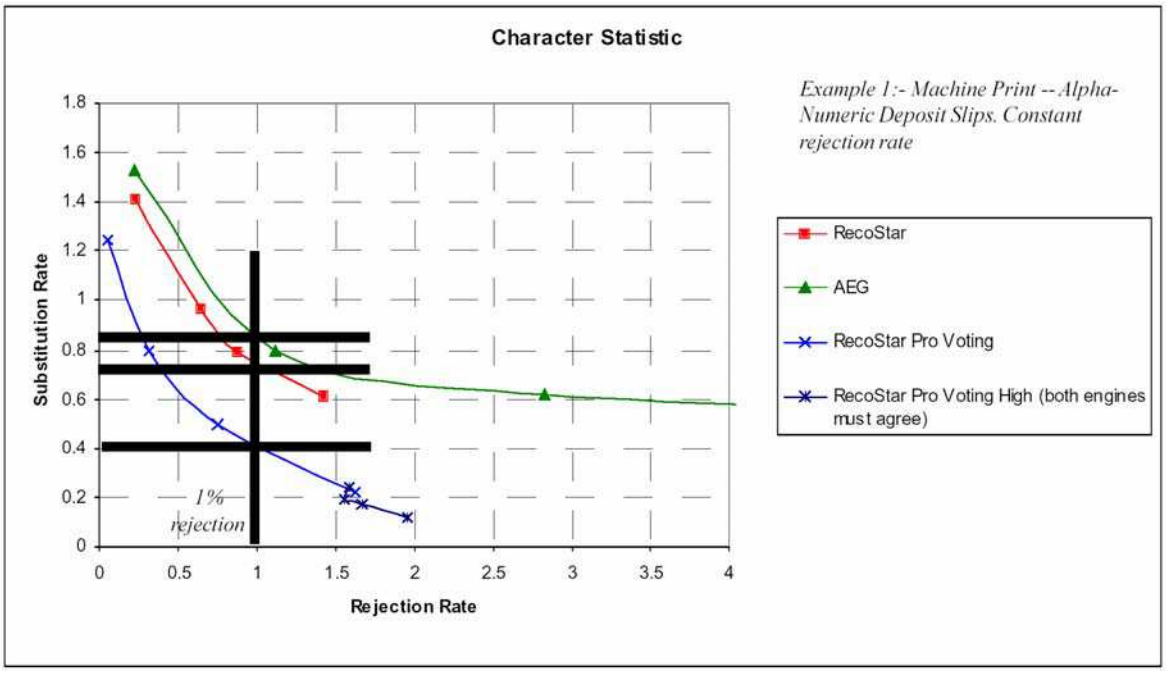


As a result of this type of internal voting, segmentation problems, which are the most costly to identify and fix, are nearly eliminated.

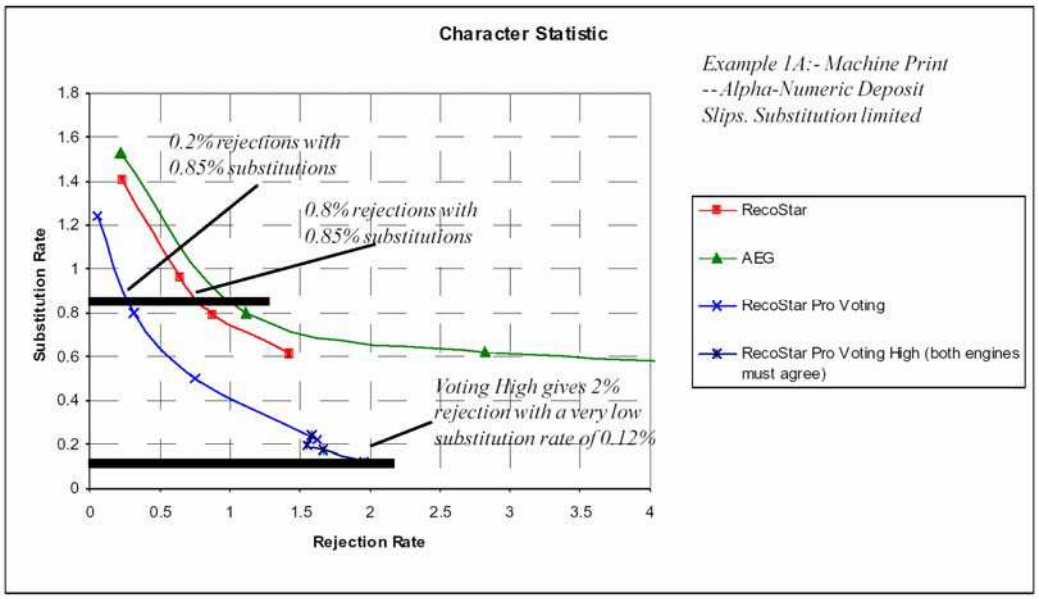
Voting systems, such as the one implemented by forms processing vendors, reduce expensive errors. If the voting engine has access to the internal OCR processes, it can make the fine adjustments in its iterative process that are needed to reduce substitutions on the most problematic characters. This type of internal voting substantially improves the value of voting as can be easily seen in the following examples.

Results of using RecoStar Voting with Real Life Examples

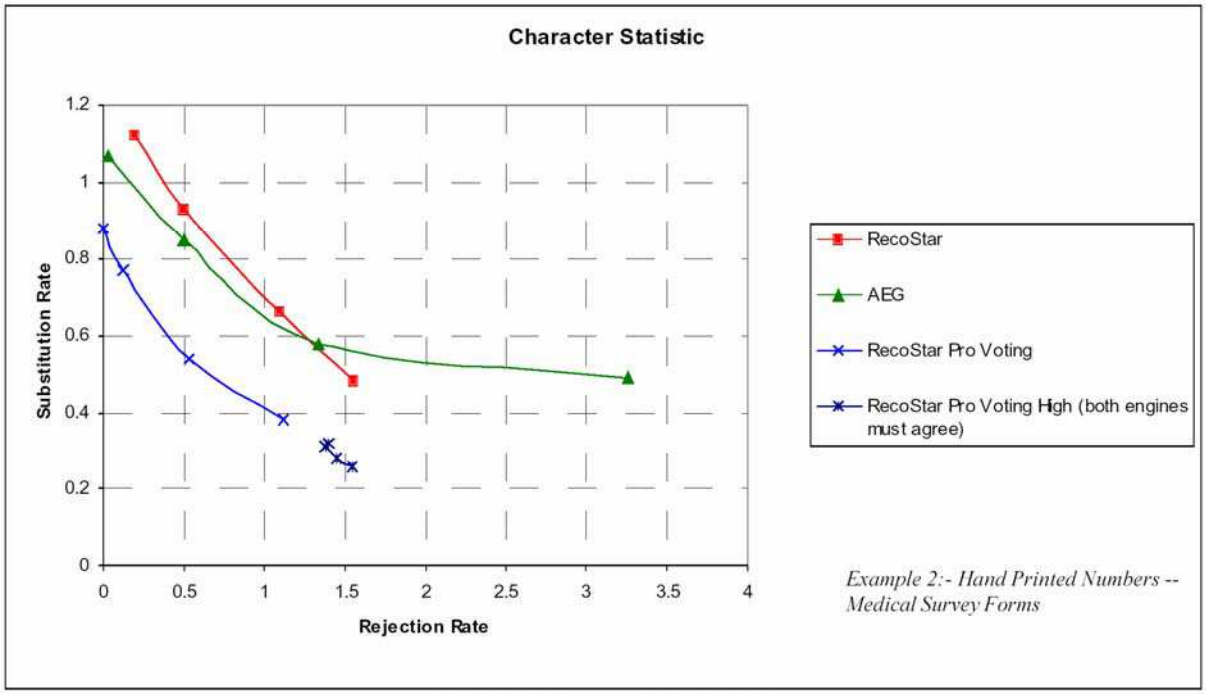
As a higher level of accuracy is sought with less substitutions, the numbers of rejected characters will increase. The charts below compare the single AEG and RecoStar products with the voting engine "RecoStar Professional". As can be clearly seen, the number of errors is consistently substantially less when using the RecoStar Professional than with either of the single engines.



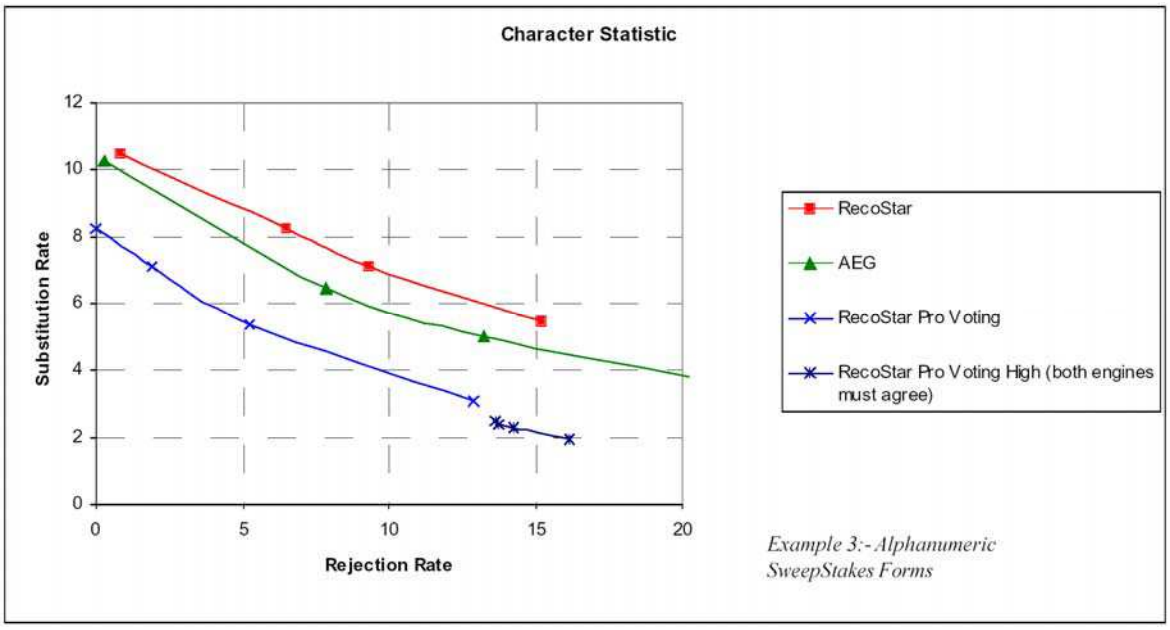
With a rejection rate of 1%. the single AEG engine creates 0.85% substitutions and the RecStar engine creates 0.75% substitutions, but the voting RecoStar Pro Voting engine creates only 0.4% substitutions.



Alternatively, with .85% substitutions, the single AEG engine rejects 1%, the RecoStar engine rejects 0.8%, while the Voting RecoStar Pro engine rejects 0.2%. For important fields. the RecoStar Pro engine can be set to agree on both engines. In this case substitutions are nearly eliminated (0.12%). but more characters are rejected (2%).



Example 2:- Hand Printed Numbers -- Medical Survey Forms



Example 3:- Alphanumeric SweepStakes Forms

Legend:

Substitution Rate (SR):

Percentage of all characters with confidence levels above acceptance threshold but wrongly recognized. It is also know as "false positive rate" or error rate. This percentage determines the quality of an OCR engine as these results cannot be corrected unless some data validation rules are applied.

Rejection Rate (RR):

Percentage of all characters with confidence levels below acceptance threshold regardless whether the result is correct or incorrect. These characters are usually displayed on a keying station for verification.

Accuracy Rate (AR):

Percentage of all characters with confidence levels above acceptance threshold and correctly recognized. This percentage rate is only implied and has to be determined by the formula as follows:

$$AR = 100\ \% - SR - RR$$

How to read the charts:

The charts on the previous pages show real life examples of single engines and voting.

In the application shown in Example 1, the single engine AEG recognizes 98.15% of the data correctly. The substitution rate is 0.85% with a rejection rate of 1%. The single RecoStar engine has a accuracy rate of 98.25%, substitution rate of 0.75% and the same rejection rate of 1%.

Using the voting engine RecoStar Pro with the same rejection rate of 1%, the substitution rate drops to 0.4%. In other words, the error rate has been cut by half.

If the error rate of the single engine is acceptable, the voting engine can be used to reduce the rejection rate. Consider the following as shown on Page 12. At 1% rejection rate, AEG has an error rate of 0.85%. Recognizing the same application with the voting engine and keeping the same error rate of 0.85%, the rejection rate drops from 1% down to 0.25%. In other words, the amount of data to be keyed or to be verified by a data entry person will be cut by 75%.