# Recognizing Garbage in OCR Output on Historical Documents

Richard Wudtke
CIS – University of Munich
wudtke@cis.uni-muenchen.de

Christoph Ringlstetter
CIS – University of Munich
kristof@cis.uni-muenchen.de

Klaus U. Schulz
CIS – University of Munich
schulz@cis.uni-muenchen.de

## ABSTRACT
Erroneous tokens in the output of an OCR engine can be roughly divided into two categories. For less serious OCR errors typically human readers - in many cases also text correction systems - are able to reconstruct the correct original word, or to suggest a small set of plausible corrections. Sometimes, however, the OCR output contains "garbage" output tokens for which it is completely impossible to predict the correct word. Garbage tokens are for example caused by graphics occurring in images misinterpreted as text by the OCR engine. In this paper we report on the development of a classifier for garbage tokens in OCR output on historical documents. The classifier is based on a specific feature set and implemented as a support vector machine. In our experiments it clearly outperformed simple rule-based predecessor solutions for OCR garbage detection.

## 1. INTRODUCTION
In this paper we face a situation where we apply an engine for optical character recognition (OCR) to a difficult text image. Typically the OCR output then contains erroneous tokens [Kuk92, RNN99]. The errors introduced by the OCR engine can be inessential or severe. In the extreme case, even human readers of the OCR output cannot make any reasonable prediction of the underlying correct word of the ground truth text. In this paper, tokens of this form are called *OCR garbage*. OCR garbage tokens may correspond to a sequence of words in the text, to only a part of a word, or even to non-textual elements (graphics) of the image that are misinterpreted as text by the OCR engine.

The automated recognition of OCR garbage tokens has several interesting applications. In automated postcorrection of OCR results [TBC, DHH$^{+}$97, RSM07] we better do not introduce any suggestion for correcting garbage tokens. In an information retrieval scenario [GRR$^{+}$11] these tokens should be excluded in order to avoid useless index entries; note that it might make sense to index non-severe errors and to use some form of approximate search during retrieval. In interactive postcorrection it makes sense to treat garbage tokens in a special workflow. Obviously most users will make a distinction between garbage tokens on the one hand and tokens where the postcorrection system offers meaningful and plausible correction suggestions on the other hand. Finally, libraries that receive large bunches of OCRed material, say, by Google [Vin07] or some service provider, are interested in ways of how to check the percentage of garbage contained in the OCRed material as a simple form of quality control.

A simple rule-based system for recognizing OCR garbage has been introduced by Taghva [TNCB01]. The main focus in [TNCB01] was the removal of strings that result when the OCR system tries to interpret graphics as text. Later Kulp and Kontostathis [KK07] suggested an improved version. The main motivation was reducing the size of the index in information retrieval applications. In [KK07], the notion of "garbage" appears to include all OCR errors. In this paper we look at the problem of how to find garbage tokens in the above restricted sense in the context of processing historical documents. In this special situation, the recognition of OCR errors, the recognition of garbage and the correction of non-severe OCR errors is even more difficult due to non-standardized orthography of the language of the texts [GRRS09]. After studying a large development set of OCR errors - both containing garbage and non-severe errors - we created via manual inspection a list of features on token-level, some of them language specific, that allow us to recognize garbage tokens. We then trained a support vector machine [Bur98] with positive and negative (correct tokens or non-severe errors) examples of garbage. The resulting classifier was evaluated in a series of experiments. Our results show that the classifier leads to a considerable improvement compared to the use of the afore-mentioned rule-based systems.

The paper has the following structure. In Section 2 we first represent typical real-life examples for OCR garbage and non-severe OCR errors we found in our work on OCR of historical documents. In Section 3 we briefly describe the two rule-based systems mentioned above. In Section 4 we describe the corpora we collected for training and evaluation of our classifier for OCR garbage. Section 5 describes the system of features that has been created by manual inspection of the development data. This system may be seen as the central contribution. Section 6 briefly describes our use of support vector machines. In Section 7 we describe our experimental results which show the new classifier outperforms the rule-based systems. In the Conclusion (Section 8) we add a brief resume.

## 2. GARBAGE IN REAL OCR OUTPUT

Consider the following lines, received from a high-level commercial OCR engine on a challenging historical German text[1] of the Bavarian State Library:

```
     M tan aber kein bessere nuten vttttda~umeye
fit Wfer $u%itftrf%km QmbfymUm/alsdie der
```

The two lines contain erroneous tokens where even readers that are experts for historical German cannot make a reasonable guess on the underlying correct word. Let us agree that by a *garbage token* we mean an erroneous OCR token where it is impossible to predict the ground truth text corresponding to the *full* token just seeing the OCR token in isolation. Interestingly, even readers not familiar with German at all have an excellent chance to correctly guess that the tokens "`vttttda~umeye`" and "`$u%itftrf%km`" are garbage in this sense. Readers not familiar with historical German, on the other hand, will probably have difficulties to judge if also "`QmbfymUm/alsdie`" represents garbage in the above sense, which is the common opinion of our experts on historical German.

Clearly, it is not possible to give a fully formal and precise definition of "OCR garbage". Still, using the above informal "definition" we found that distinct readers in most cases agree in their judgements when asked to classify given tokens into "garbage" versus "non-garbage". This can be seen as a justification for the enterprise to develop and evaluate an automated classifier. Some real-life examples for garbage tokens found in a collection of OCR output documents for German historical texts are `gtbfftre, NnÄLwig-MMen:, 0--M----H>, $♯♯.ft♯gÄ, mowMmtfttzm`. Borderline cases found are, e.g. `d!ttt, Fe?6icht/, Tsyl, fttr`. An example with images and OCR output aligned that shows garbage and non-garbage tokens can be found in Figure 1. As a matter of fact, it is often difficult to classify short words. This is due to the effect that we typically obtain a considerable amount of possible corrections even using only a small number of edit operations as correction steps.

## 3. TWO RULE-BASED SYSTEMS FOR DETECTION OF OCR GARBAGE

In 2001, Taghva et al. [TNCB01] introduced a rule-based algorithm for garbage detection as a tool for the MANICURE text correction system [TCB$^+$98]. The algorithm is based on 6 simple and independent rules, each characterizing a class of strings to be considered as garbage.

1. A string composed of more than 40 symbols is garbage.
2. A string that contains more special symbols than letters and digits is garbage.

[1]The example document is challenging in three ways: printed in Gothic font, written in historical German from the Early New High German period and with a high amount of noise in the scanned image originating from the poor state of the hard copy.

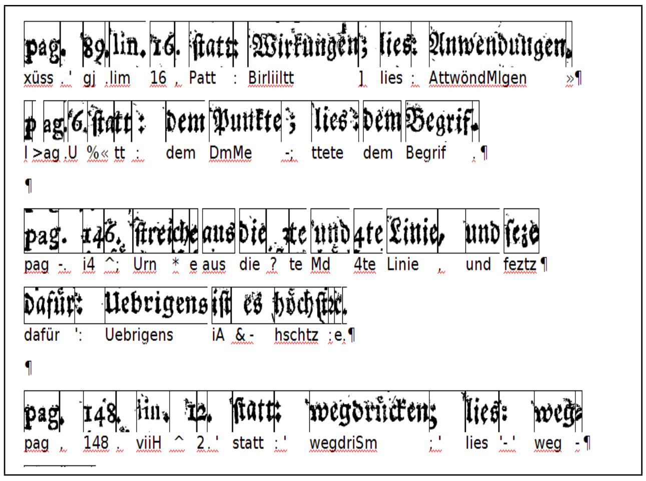
xüss .' gj .lim 16 , Patt : Birliiltt ] lies : AttwöndMlgen »¶
I >ag .U %« tt : dem DmMe -; ttete dem Begrif , ¶
¶
pag -. i4 ^; Urn * e aus die ? te Md 4te Linie , und feztz ¶
dafür ': Uebrigens iA & - hschtz ;e.¶
¶
pag , 148 . viiH ^ 2.' statt :' wegdriSm ;' lies '-' weg -¶

**Figure 1: Images and OCR result aligned showing garbage and non-garbage tokens.**

3. A string containing at least 4 consecutive occurrences of the same symbol is garbage.
4. If the string is only composed of vowels and consonants and the the number of consonants is greater than 10 times the number of vowels, or vice versa, the string is garbage (variant of 1).
5. When we delete both the first and the last symbol of a string and the rest contains at least two distinct punctuation symbols, then the string is garbage.
6. If a string starts and ends with lower-case letters and contains an upper-case letter it is garbage.

The main interest in [TNCB01] was the detection of strings generated by the OCR engine when misinterpreting graphics as text (cf. Figure 2). In order to improve the recall of garbage detection, Kulp and Kontostathis [KK07] modified the rule set. They used rules, 5, 6 and in addition the following rules.

7. A string composed of more than 20 symbols is garbage (variant of 1).
8. A string containing at least 3 consecutive occurrences of the same symbol is garbage (variant of 3).
9. If the number of upper-case letters is larger than the number of lower-case letters and the string contains both types of letters it is garbage.
10. If a string is only composed of vowels and consonants and the the number of consonants is greater than 8 times the number of vocals, or vice versa, the string is garbage (variant of 4).
11. If a string contains 4 consecutive vowels or 5 consecutive consonants it is garbage.

In difference to Taghva et al. [TNCB01] Kulp and Kontostathis [KK07] were interested in detection of arbitrary OCR errors. This explains the use of a more restrictive rule set.

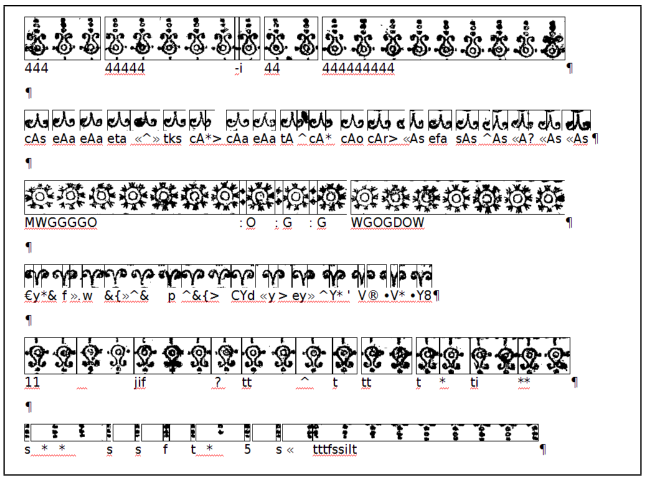
444 44444 -i 44 444444444 ¶
¶
cAs eAa eAa eta «^» tks cA*> cAa eAa tA ^cA* cAo cAr> «As efa sAs ^As «A? «As «As ¶
¶
MWGGGGO :O :G :G WGOGDOW ¶
¶
€y*& f».w &{»^& p ^&{> CYd «y> ey» ^Y*' V® •V* •Y8¶
¶
11 jif ? tt ^ t tt t * ti ** ¶
¶
s * * s s f t * 5 s« tttfssilt ¶

**Figure 2: Graphical elements recognized as text by the OCR, resulting in garbage output.**

## 4. CORPUS MATERIALS FOR TRAINING AND EVALUATION

The present work was done in the context of the EU project IMPACT [IMP]. The principal goal of IMPACT is the improvement of OCR for mass digitization of historical documents. The interest in methods for garbage removal resulted when we started to implement and test a system for interactive postcorrection of OCR-results. For difficult (regions of) documents we sometimes found a considerable amount of garbage in the output. In the final system we predict a positive effect to organize such tokens in a separate workflow distinct from those where reasonable correction candidates are available.

In our experiments we used a corpus composed of 15 distinct historical German documents from the Bavarian State Library, with 5 documents from the 16th, 17th, and 18th century, respectively. The total number of tokens in the corpus is 143,860. The documents were processed using a commercial OCR engine with special support for the recognition of German historical documents in Gothic font (Abbyy Finereader Version 10). From the resulting OCR output we created two lists of approximately the same size, one containing garbage tokens, the other containing correct words or tokens representing non-severe OCR errors. At this point we ignored problems caused by wrong word segmentation: all OCR tokens that represent merges or splits of tokens of the ground truth text were excluded from the data. The idea was to consider these cases later in a separate step. The total number of tokens in both lists is 6,395. We used 80% of the annotated material for training (feature detection, training of a garbage classifier) and 20% for evaluation. Details are described in the next section, together with a second evaluation corpus composed of running text.

## 5. FEATURES FOR GARBAGE DETECTION

For deriving appropriate features, the training parts of the two lists mentioned in the previous section were manually inspected and features were selected on the basis of pretests which measured how selective they performed on the training set. For example, we measured the average length of the strings in the training examples of the garbage and the non-garbage sets. As a matter of fact, also the rules sets described in Section 3 were taken into account. Simple features used in our system - more or less directly inspired by the above rules - are the following:

(1) the length $l$ of the input string,

(2) the number $v$ of vowels and the number $c$ of consonants in the string, as well as the quotients $v/l$, $c/l$, $v/c$ (for $c \neq 0$),

(3) the number of special (non-alphanumerical) symbols $s$ and the quotient $s/l$,

(4) the number of digits $d$ and the quotient $d/l$,

(5) the number of lowercase letters *low*, the number of uppercase letters *upp*, and the quotients $low/l$, $upp/l$,

(6) for strings containing a sequence of at least three consecutive occurrences of the same symbol, we use the quotient of the length of the maximal sequence of identical letters divided by $l$. For other strings the feature receives value 0.

(7) We calculate the number of all alpha-numerical symbols $l_\alpha$ occurring in the string, and the number $k$ of other symbols $s$. For $k > l_\alpha$ the value Feature 7 is 1, for other strings the value is 0.

(8) If the input string contains a subsequence of $\geq 6$ directly consecutive consonants, Feature 8 received value 1, otherwise value 0.

(9) We delete the first and last symbol of the input string. If the remaining infix contained two ore more non-alpha-numerical symbols, Feature 9 receives value 1, and otherwise value 0.

We treated the letter "v" as a vowel since in old German texts it typically acts as a substitute for modern "u". In addition to Features 1-9 we used the following four features - two of them language specific.

*Bigram.* From a large corpus of historical German documents with 3,552,690 tokens [GRR+11] we compiled a frequency list $L_B$ of letter bigrams occurring in (correct) historical German words in an offline step. We computed the list of bigrams for the given input string, which was normalized to lowercase. For each bigram we checked if it occurs in the list $L_B$ and we looked at its frequency value. From these numbers we derived a value for the total "naturalness" of bigrams in the string. Formally, this value is defined as

$$bigr := \frac{\sum_{i=1}^{n} bf_i/c}{n}$$

where $n$ is the number of bigrams in the input string, $bf_i$ denotes the frequency of the $i$-th bigram in the list $L_B$, and

$c = 10,000$ is a scaling constant. In tests on the training materials we found that the average value for garbage strings is 5.3 while the average value for non-garbage strings is 17.4.

*Most frequent symbol.* We computed the number of occurrences $i$ of the most frequent symbol of the input string of length $l$. For $i \geq 3$ we used $i/l$ as a feature value, for $i \leq 2$ the value was set to 0. The feature is motivated by the observation that often garbage strings contain many occurrences of the same symbol.

*Non-alphabetical symbols.* Let $l_1$ denote the number of occurrences of alphabetical symbols in the input string, let $l_2 = l - l_1$ denote the number of occurrences of all other types of symbols. We used $l_2/l_1$ as a feature.

*Levenshtein distance.* Our most complex feature tries to measure the plausibility of finding a simple (possibly vacuous) correction of the input string to a "plausible historical word". This feature involves some subtlety: having no historical lexicon at our disposal we needed to establish a link between a potentially noisy historical input string and a modern lexicon to compute correction candidates. The problem is solved in a two stage process. As a result of our work in the EU project IMPACT we derived a set of 140 "patterns" (re-write rules) that explain the typical differences between modern and old spelling for German. For example, the pattern $t \mapsto th$ explains the difference between modern word "Turm" (tower) and the spelling "Thurm" often found in historical German documents. Let us say that a string $U$ represents a *possible historical word* if it can be derived from a modern word $W$ using a (possibly empty) sequence of parallel pattern applications. We say that an input string $V$ has a *plausible correction* (to a possible historical string) if there exists a possible historical word $U$ such that the Levenshtein distance between $V$ and $U$ does not exceed a given bound. Reffle in [Ref11] introduced a tool that efficiently checks if an input string $V$ has a plausible correction and generates all plausible corrections.[2] Both the maximal number of pattern applications (between a modern word and a possible historical word) and the bound for the Levenshtein distance (between input string and possible historical word) can be selected by the user. In experiments we found that 80% of garbage words in our list do *not* have any plausible correction, while the corresponding value for non-garbage words is only 6.8%. We choose $b = 2$ as the maximal Levenshtein distance. Our Levenshtein feature has the value $l$ (length of input string $V$) if $V$ does not have a plausible correction. In the other case the value is $(\nu + p/2 + 1)/l$ where $\nu$ is the minimal Levenshtein distance between $V$ and a plausible correction $U$ and $p$ is the number of pattern applications needed to derive $U$ from a modern word $W$.

Let us remark that in particular the last feature is clearly tailored towards recognition of garbage in the strict sense (severe OCR errors). Obviously, if we wanted to recognize *any* OCR error, it would make more sense just to check if the given input string *is* a possible historical word.

[2]Lacking the highly efficient implementation a practical application of the feature would not be feasible because a naive algorithm would have to check each string of the input document against the whole lexicon. For details see [Ref11].

| | SVM-Lev | SVM+Lev | T. et al. | K. & K. |
|---|---|---|---|---|
| Accuracy | 87.55% | 91.07% | 66.95% | 68.75% |
| Precision | 91.14% | 89,98% | 90.15% | 80.15% |
| Recall | 83.33% | 92.51% | 38.47% | 50.13% |

**Table 1: First series of experiments: Classifying strings from two given lists (respectively containing garbage strings and non-garbage strings) with four methods. Column SVM-Lev (resp. SVM+Lev) refers to the classifier where the Levenshtein feature (cf. Section 5) was not used (was used).**

## 6. USE OF SUPPORT VECTOR MACHINES

To decide whether a string falls into garbage and or not defines a binary classification problem where the data points kann be seen as k-dimensional vectors, with k as the number of features defined in Section 5 . For this kind of problem support vector machines (SVM) are considered as the best choice. On the training set of human classified positive and negative examples an SVM is trained to achieve a model that divides the examples of the two categories with maximum margin. In our experiments we used the library LIBSVM [CL01]. After a series of preliminary tests, a variant of SVMs with RBF (radial basis function) kernel offered by LIBSVM was used.

## 7. EVALUATION EXPERIMENTS

In our first series of experiments the classifier was applied to a selection of 642 garbage tokens and 635 other tokens, the subsets for testing from the two lists mentioned in Section 4. In the first (second subexperiment), the classifiers were based on feature sets excluding (including) the Levenshtein feature. The afore-mentioned rule-based systems by Taghva et al. [TNCB01] and Kulp and Kontostatis [KK07] were also applied for comparison. In what follows, *accuracy* is defined as the total number of correctly classified strings divided by the total number of strings classified. *Precision* is the number of correctly classified garbage strings divided by the total number of strings classified as garbage. *Recall* is the number of correctly classified garbage strings divided by the total number of garbage strings.

Results are shown in Table 1. When interpreting these numbers it should be kept in mind that the main focus of the rule-based systems (i.e., the respective notion of "garbage") is slightly distinct from our motivation as was mentioned in Section 3. The new classifiers clearly outperform the simple rule-based systems. In the present application domain, the two rule systems in particular lead to low recall. We also see that the use of the Levenshtein feature leads to a significant gain of recall (9.18%), at the cost of only a slightly reduced precision (1.16%).

In a second series of experiments we studied the behavior of the classifiers on running text of OCR output as opposed to the sample lists of tokens used for the first experiment(Table 1). We considered 6 additional texts from the 16th century (different from those used for the training/test set of the first experiment) . From each text we processed 10 pages with OCR. From these 60 pages, 11,247 tokens (running text) of length greater or equal 4 were manually tagged

| | SVM-Lev | Taghva. et al. | Kulp & Kontostathis |
|---|---|---|---|
| Accuracy | 88.11% | 85.74% | 84.58% |
| Precision | 76.80% | 33,47% | 44.87% |
| Recall | 53.81% | 45.31% | 42.59% |
| F-Score | 63.28% | 38.50% | 43.70% |

**Table 2: Second series of experiments: classifying strings from running OCR output using three methods where merges and splits in the OCR output were not corrected.**

| | SVM-Lev | Taghva et al. | Kulp & Kontostathis |
|---|---|---|---|
| Accuracy | 88.44% | 83.02% | 82.36% |
| Precision | 84.25% | 35.86% | 45.09% |
| Recall | 67.69% | 66.54% | 59.66% |
| F-Score | 75.07% | 46.60% | 51.36% |

**Table 3: Third series of experiments: classifying strings from running OCR output using three methods where merges and splits in the OCR output were corrected.**

into garbage versus non garbage. We applied the SVM classifier where the Levenshtein feature was not used. Results are presented in Table 2. Both rule based systems classified manny tokens incorrectly as garbage which leads to very poor precision. Note that the considerably higher accuracy is attributed to the high number of non-garbage tokens in running text than in the balanced lists of the first experiment. In a third series of experiments, we proceeded in the same way, however if possible we first corrected merges and splits manually in the OCR output; otherwise they were declared to garbage. The gain in precision for the SVM classifier reflects that share of the data where only the word segmentation went wrong and that were otherwise correctly recognized, thus classified as non-garbage. The results are presented in Table 3.

## 8. CONCLUSION

We introduced a new method for classifying OCR output tokens into garbage versus non-garbage tokens. The intended application domain is OCR output received from processing historical documents. Our notion of garbage is strict in the sense that erroneous tokens are not treated as garbage if there is a real chance to reconstruct the correct word just using the OCR output. For training and evaluation experiments, a very large number of OCR output strings were manually tagged. We both looked at preselected lists of garbage/non-garbage tokens and at running text of OCR output. Our classifiers clearly outperformed simple rule-based systems known from the literature. Besides the superior weighting of the features through the SVM classifier we also saw positive effects of the consideration of language specifics. From our point of view, the strong empirical foundation of the experiments and the carefully selected set of features represent the main contribution. We think that further improvements of the feature system are possible, e.g., by refining the bigram feature.

**Acknowledgements.** The authors were supported by EU project IMPACT.

## 9. REFERENCES

[Bur98] Christopher J.C. Burges. A tutotial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[CL01] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines, http://www.csie.ntu.edu.tw/cjlin/libsvm, 2001.

[DHH+97] Andreas Dengel, Rainer Hoch, Frank Hönes, Thorsten Jäger, Michael Malburg, and Achim Weigel. Techniques for improving OCR results. In Horst Bunke and Patrick S.P. Wang, editors, *Handbook of Character Recognition and Document Image Analysis*, pages 227–258. World Scientific, 1997.

[GRR+11] Annette Gotscharek, Ulrich Reffle, Christoph Ringlstetter, Klaus Schulz, and Andreas Neumann. Towards information retrieval on historical document collections: the role of matching procedures and special lexica. *International Journal on Document Analysis and Recognition*, 14:159–171, 2011.

[GRRS09] Annette Gotscharek, Ulrich Reffle, Christoph Ringlstetter, and Klaus U. Schulz. On lexical resources for digitization of historical documents. In *DocEng '09: Proceedings of the 9th ACM symposium on Document Engineering*, pages 193–200, New York, NY, USA, 2009. ACM.

[IMP] EU project Improving Access to Text IMPACT, http://www.impact-project.eu/.

[KK07] Scott Kulp and April Kontostathis. On retrieving legal files: Shortening documents and weeding out garbage. In Ellen M. Voorhees and Lori P. Buckland, editors, *TREC*, volume Special Publication 500-274. National Institute of Standards and Technology (NIST), 2007.

[Kuk92] Karen Kukich. Techniques for automatically correcting words in texts. *ACM Computing Surveys*, pages 377–439, 1992.

[Ref11] Ulrich Reffle. Efficiently generating correction suggestions for garbled tokens of historical language. *Nat. Lang. Eng.*, 17:265–282, 2011.

[RNN99] Stephen V. Rice, George Nagy, and Thomas A. Nartker. *Optical Character Recognition: An Illustrated Guide to the Frontier*. Kluwer Academic Publishers, 1999.

[RSM07] Christoph Ringlstetter, Klaus U. Schulz, and Stoyan Mihov. Adaptive text correction with web-crawled domain-dependent dictionaries. *ACM Trans. Speech Lang. Process.*, 4(4):9, 2007.

[TBC] Kazem Taghva, Julie Borsack, and Allen Condit. An expert system for automatically correcting ocr output. Aus http://citeseer.nj.nec.com/cs ResearchIndex.

[TCB$^+$98] Kazem Taghva, Allen Condit, Julie Borsack, John Kilburg, Changshi Wu, and Jeff Gilbreth. The MANICURE document processing system. In *In Proc. IS&T/SPIE 1998 Int. Symp. on Electronic Imaging Science and Technology*, 1998.

[TNCB01] Kazem Taghva, Tom Nartker, Allen Condit, and Julie Borsack. Automatic removal of "garbage strings" in OCR text: An implementation. In *The 5th World Multi-Conference on Systemics, Cybernetics and Informatics*, 2001.

[Vin07] Luc Vincent. Google book search: Document understanding on a massive scale. In *Proceedings of the Conference on Document Analysis and Recognition (ICDAR09)*, 2007.