OCR Errors

by Michael Barz
Motivation

• In general: How to get information out of noisy input?
  – Dealing with noisy input (scan/fax/e-mail...) in written form

• Approach: Combination of diverse NLP tools in one pipeline
  – Optical Character Recognition (OCR)
  – Sentence Boundary Detection
  – Tokenization
  – Part-of-Speech Tagging

• Efficient evaluation method for OCR results (from pipeline)
  – Dynamic programming approaches \(\rightarrow\) mathematical description
  – Error identification (where does the error come from?)

• Techniques to improve pipeline (avoid errors)
  – Table spotting
Pipeline

Noisy Input

Optical Character Recognition (OCR) → Sentence Boundary Detection → Tokenization → Part-of-Speech Tagging → Result
Noisy Input

Optical Character Recognition (OCR) → Sentence Boundary Detection → Tokenization → Part-of-Speech Tagging → Result
Noisy Input

Clean

Noisy

**ABSTRACT**

Errors are unavoidable in advanced computer vision applications such as optical character recognition, and the noise induced by these errors presents a serious challenge to downstream processes that attempt to make use of such data. In this paper, we apply a new paradigm we have proposed for measuring the impact of recognition errors on the stages of a standard text analysis pipeline: sentence boundary detection, tokenization, and part-of-speech tagging. Our methodology formulates error classification as an optimization problem solvable using a hierarchical dynamic programming approach. Errors and their cascading effects are isolated and analyzed as they travel through the pipeline. We present experimental results based on a large collection of scanned pages to study the varying impact depending on the nature of the error and the character(s) involved. The problem of identifying tabular structures that should not be parsed as sentential text is also discussed.
Noisy Input

• Generating noisy input to test pipeline
  – Printed digital writing
  – Scanned directly for clean input
  – Repeated copies combined with fax
    → noisy input
Optical Character Recognition

- Optical Character Recognition (OCR)
- Sentence Boundary Detection
- Tokenization
- Part-of-Speech Tagging

Result
Optical Character Recognition

• “Conversion of the scanned input image from bitmap format to encoded text”

• Possible Errors (impact on later stages)
  – Punctuation errors
  – Substitution errors
  – Space deletion

• Tools: gocr, Tesseract
Sentence Boundary Detection

Noisy Input

Optical Character Recognition (OCR) → Sentence Boundary Detection → Tokenization → Part-of-Speech Tagging → Result
Sentence Boundary Detection

• “break the input text into sentence-sized units, one per line”
• Usage of syntactic (and semantic) information
• Tool: MXTERMINATOR
Tokenization

Noisy Input

Optical Character Recognition (OCR)

Sentence Boundary Detection

Tokenization

Part-of-Speech Tagging

Result
Tokenization

• “breaks it into individual tokens which are delimited by whitespace”
  – Tokens: words, punctuation symbols

• Tool: Penn Treebank tokenizer
Part-of-Speech Tagging

1. Optical Character Recognition (OCR)
2. Sentence Boundary Detection
3. Tokenization
4. Part-of-Speech Tagging

Result
Part-of-Speech Tagging

- Assigns meta information to tokens due to their part of speech
- Tool: MXPOST
Sample Result

Ground-Truth

<table>
<thead>
<tr>
<th>VB</th>
<th>IN</th>
<th>DT</th>
<th>NNS</th>
<th>IN</th>
<th>NNS</th>
<th>RB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look</td>
<td>at</td>
<td>the</td>
<td>crowds</td>
<td>of</td>
<td>water-gazers</td>
<td>there</td>
</tr>
</tbody>
</table>

OCR Output

<table>
<thead>
<tr>
<th>NN</th>
<th>IN</th>
<th>DT</th>
<th>NNS</th>
<th>IN</th>
<th>NNS</th>
<th>RB</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>oo</em></td>
<td>at</td>
<td>the</td>
<td>, rows</td>
<td>of</td>
<td>water-gazers</td>
<td>th_re</td>
</tr>
</tbody>
</table>
Why evaluation?

• Errors occur
  – Propagate through stages of pipeline
  – Different types (as mentioned at OCR)

• Which impact do errors have?
Performance Evaluation

- Dynamic programming approach
- Levenshtein distance for each stage (adjusted)
- Compare part-of-speech tags after
- Try to backtrack where errors arise and which impact they have
Performance Evaluation

\[
dist_{1i,j} = \min \left\{ \begin{array}{l}
dist_{1i-1,j} + c1_{del}(s_i) \\
dist_{1i,j-1} + c1_{ins}(t_j) \\
dist_{1i-1,j-1} + c1_{sub}(s_i, t_j) \\
\end{array} \right. 
\]

<table>
<thead>
<tr>
<th></th>
<th>e</th>
<th>T</th>
<th>o</th>
<th>r</th>
</tr>
</thead>
<tbody>
<tr>
<td>e</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>T</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>i</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>r</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>
Performance Evaluation

- Extention: Substitution of more than one sign

\[
\text{dist}1_{i,j} = \min \begin{cases} 
\text{dist}1_{i-1,j} + c_1 \text{del}(s_i) \\
\text{dist}1_{i,j-1} + c_1 \text{ins}(t_j) \\
\min_{1 \leq k' \leq k, \ 1 \leq l' \leq l} \left[ \text{dist}1_{i-k',j-l'} + \\
c_1 \text{sub}_{k,l}(s_{i-k'+1...i}, t_{j-l'+1...j}) \right] 
\end{cases}
\]
Performance Evaluation

Token-Distance (dist2)
- Costs for inserting, deleting or substituting a token are defined as
  - dist1(ε, t)
  - dist1(s, ε)
  - Distance between substituted substrings

Sentence-Distance (dist3)
- Costs for inserting, deleting or substituting a sentence are defined as
  - dist2(ε, t)
  - dist2(s, ε)
  - Distance between substituted tokens
**Table 1: Average OCR performance relative to ground-truth.**

<table>
<thead>
<tr>
<th></th>
<th>All Symbols</th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec.</td>
<td>Recall</td>
<td>Overall</td>
<td>Prec.</td>
<td>Recall</td>
</tr>
<tr>
<td>Clean</td>
<td>0.982</td>
<td>0.988</td>
<td>0.988</td>
<td>0.843</td>
<td>0.973</td>
</tr>
<tr>
<td>Light</td>
<td>0.646</td>
<td>0.747</td>
<td>0.790</td>
<td>0.193</td>
<td>0.648</td>
</tr>
<tr>
<td>Dark</td>
<td>0.411</td>
<td>0.575</td>
<td>0.628</td>
<td>0.090</td>
<td>0.686</td>
</tr>
<tr>
<td>Fax</td>
<td>0.584</td>
<td>0.644</td>
<td>0.732</td>
<td>0.201</td>
<td>0.625</td>
</tr>
<tr>
<td></td>
<td>Whitenspace</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Prec.</td>
<td>Recall</td>
<td>Overall</td>
<td>Prec.</td>
<td>Recall</td>
</tr>
<tr>
<td>Clean</td>
<td>0.974</td>
<td>0.996</td>
<td>0.985</td>
<td>0.589</td>
<td>0.965</td>
</tr>
<tr>
<td>Light</td>
<td>0.391</td>
<td>0.884</td>
<td>0.539</td>
<td>0.608</td>
<td>0.890</td>
</tr>
</tbody>
</table>

---

**Table 2: Average text processing performance relative to ground-truth.**

<table>
<thead>
<tr>
<th></th>
<th>Sentence Boundaries</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec.</td>
<td>Recall</td>
<td>Overall</td>
<td>Prec.</td>
<td>Recall</td>
<td>Overall</td>
<td>Prec.</td>
<td>Recall</td>
<td>Overall</td>
<td></td>
</tr>
<tr>
<td>Clean</td>
<td>0.939</td>
<td>0.985</td>
<td>0.961</td>
<td>0.975</td>
<td>0.994</td>
<td>0.984</td>
<td>0.953</td>
<td>0.975</td>
<td>0.964</td>
<td></td>
</tr>
<tr>
<td>Light</td>
<td>0.648</td>
<td>0.906</td>
<td>0.731</td>
<td>0.646</td>
<td>0.877</td>
<td>0.733</td>
<td>0.307</td>
<td>0.500</td>
<td>0.380</td>
<td></td>
</tr>
<tr>
<td>Dark</td>
<td>0.321</td>
<td>0.995</td>
<td>0.405</td>
<td>0.388</td>
<td>0.691</td>
<td>0.479</td>
<td>0.097</td>
<td>0.210</td>
<td>0.132</td>
<td></td>
</tr>
<tr>
<td>Fax</td>
<td>0.442</td>
<td>0.987</td>
<td>0.536</td>
<td>0.494</td>
<td>0.674</td>
<td>0.563</td>
<td>0.203</td>
<td>0.303</td>
<td>0.242</td>
<td></td>
</tr>
</tbody>
</table>
Improve pipeline

• Tables are no sentences → Pipeline won’t work well
• Don’t regard Tables → We need an algorithm to find and spot all tables
### Table Spotting

<table>
<thead>
<tr>
<th>outside spaces</th>
<th>inside spaces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lucent 98 1/16</td>
<td>97 1/4</td>
</tr>
<tr>
<td>+ 1 1/4</td>
<td></td>
</tr>
<tr>
<td>Ascend 72 7/8</td>
<td>72 1/2</td>
</tr>
<tr>
<td>+ 1 1/8</td>
<td></td>
</tr>
</tbody>
</table>

\[ Incorr_{ws} = 22 \]

For example:

<table>
<thead>
<tr>
<th>SHORT MESSAGE</th>
<th>SOLUTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>- - - Among the new products Lucent is announcing during Wireless ’99 are an intelligent network</td>
<td></td>
</tr>
</tbody>
</table>

\[ Incorr_{ws} = -16 \]

- \( acorr(\alpha, \beta) = \begin{cases} 
1 & \text{if } \alpha \text{ and } \beta \text{ are both inside spaces} \\
-1 & \text{if one of } \alpha \text{ or } \beta \text{ is an inside space} \\
0 & \text{otherwise} 
\end{cases} \]

\[ Incorr_{ws}(i, j) = \sum_{k=1}^{m} acorr(atext[i, k], atext[j, k]) \]
Table Spotting

\[ lncorr_{ws}(i, j) = \sum_{k=1}^{m} acorr(atext[i, k], atext[j, k]) \]

\[ merit_{pre}(i, [i+1, j]) = \sum_{k=i+1}^{j} \frac{1}{e^{\gamma(k-i-1)}} \cdot lncorr_{ws}(i, k) \]

and

\[ merit_{app}([i, j-1], j) = \sum_{k=i}^{j-1} \frac{1}{e^{\gamma(j-1-k)}} \cdot lncorr_{ws}(k, j) \]

\[ tab[i, j] = \max \left\{ merit_{pre}(i, [i+1, j]) + tab[i+1, j], \quad tab[i, j-1] + merit_{app}([i, j-1], j) \right\} \]
Table Spotting

\[
tab[i, j] = \max \begin{cases} 
\text{merit}_{pre}(i, [i + 1, j]) + \text{tab}[i + 1, j] \\
\text{tab}[i, j - 1] + \text{merit}_{app}([i, j - 1], j) 
\end{cases}
\]

\[
\text{score}[i, j] = \max \begin{cases} 
\text{tab}[i, j] & 1 \leq i < j \leq n \\
\max_{i \leq k < j} \{\text{score}[i, k] + \text{score}[k + 1, j]\} 
\end{cases}
\]
Evaluation 2008

Table 1: Average OCR performance relative to ground-truth.

<table>
<thead>
<tr>
<th></th>
<th>All Symbols</th>
<th></th>
<th></th>
<th></th>
<th>Punctuation</th>
<th></th>
<th></th>
<th></th>
<th>Whitespace</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec.</td>
<td>Recall</td>
<td>Overall</td>
<td></td>
<td>Prec.</td>
<td>Recall</td>
<td>Overall</td>
<td></td>
<td>Prec.</td>
<td>Recall</td>
<td>Overall</td>
</tr>
<tr>
<td>Clean</td>
<td>0.995</td>
<td>0.997</td>
<td>0.997</td>
<td></td>
<td>0.981</td>
<td>0.996</td>
<td>0.988</td>
<td></td>
<td>0.995</td>
<td>0.999</td>
<td>0.997</td>
</tr>
<tr>
<td>Dark1</td>
<td>0.989</td>
<td>0.996</td>
<td>0.994</td>
<td></td>
<td>0.937</td>
<td>0.992</td>
<td>0.963</td>
<td></td>
<td>0.980</td>
<td>0.998</td>
<td>0.989</td>
</tr>
<tr>
<td>Dark2</td>
<td>0.966</td>
<td>0.990</td>
<td>0.981</td>
<td></td>
<td>0.797</td>
<td>0.972</td>
<td>0.874</td>
<td></td>
<td>0.929</td>
<td>0.988</td>
<td>0.958</td>
</tr>
<tr>
<td>Light1</td>
<td>0.995</td>
<td>0.997</td>
<td>0.997</td>
<td></td>
<td>0.977</td>
<td>0.994</td>
<td>0.986</td>
<td></td>
<td>0.993</td>
<td>0.999</td>
<td>0.996</td>
</tr>
<tr>
<td>Light2</td>
<td>0.994</td>
<td>0.997</td>
<td>0.997</td>
<td></td>
<td>0.971</td>
<td>0.989</td>
<td>0.981</td>
<td></td>
<td>0.992</td>
<td>0.999</td>
<td>0.996</td>
</tr>
<tr>
<td>Overall</td>
<td>0.988</td>
<td>0.995</td>
<td>0.993</td>
<td></td>
<td>0.933</td>
<td>0.989</td>
<td>0.958</td>
<td></td>
<td>0.978</td>
<td>0.997</td>
<td>0.987</td>
</tr>
</tbody>
</table>

Table 2: Average NLP performance relative to ground-truth.

<table>
<thead>
<tr>
<th></th>
<th>Sentence Boundaries</th>
<th></th>
<th></th>
<th></th>
<th>Tokenization</th>
<th></th>
<th></th>
<th></th>
<th>Part-of-Speech Tagging</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec.</td>
<td>Recall</td>
<td>Overall</td>
<td></td>
<td>Prec.</td>
<td>Recall</td>
<td>Overall</td>
<td></td>
<td>Prec.</td>
<td>Recall</td>
<td>Overall</td>
</tr>
<tr>
<td>Clean</td>
<td>0.978</td>
<td>0.995</td>
<td>0.985</td>
<td></td>
<td>0.994</td>
<td>0.997</td>
<td>0.995</td>
<td></td>
<td>0.988</td>
<td>0.991</td>
<td>0.989</td>
</tr>
<tr>
<td>Dark1</td>
<td>0.918</td>
<td>0.988</td>
<td>0.946</td>
<td></td>
<td>0.977</td>
<td>0.987</td>
<td>0.982</td>
<td></td>
<td>0.964</td>
<td>0.976</td>
<td>0.970</td>
</tr>
<tr>
<td>Dark2</td>
<td>0.782</td>
<td>0.963</td>
<td>0.850</td>
<td></td>
<td>0.919</td>
<td>0.946</td>
<td>0.932</td>
<td></td>
<td>0.885</td>
<td>0.917</td>
<td>0.900</td>
</tr>
<tr>
<td>Light1</td>
<td>0.971</td>
<td>0.994</td>
<td>0.981</td>
<td></td>
<td>0.992</td>
<td>0.996</td>
<td>0.994</td>
<td></td>
<td>0.985</td>
<td>0.989</td>
<td>0.987</td>
</tr>
<tr>
<td>Light2</td>
<td>0.967</td>
<td>0.984</td>
<td>0.972</td>
<td></td>
<td>0.990</td>
<td>0.994</td>
<td>0.992</td>
<td></td>
<td>0.983</td>
<td>0.987</td>
<td>0.985</td>
</tr>
<tr>
<td>Overall</td>
<td>0.923</td>
<td>0.985</td>
<td>0.947</td>
<td></td>
<td>0.974</td>
<td>0.984</td>
<td>0.979</td>
<td></td>
<td>0.961</td>
<td>0.972</td>
<td>0.966</td>
</tr>
</tbody>
</table>
Error identification

Figure 8: Tokenization accuracy as a function of OCR accuracy.
QUESTIONS?
Sources:
“Performance Evaluation for Text Processing of Noisy Inputs” (Daniel Lopresti, 2005)
“Optical Character Recognition Errors and Their Effects on Natural Language Processing” (Daniel Lopresti, 2009)

THANK YOU FOR YOUR ATTENTION!