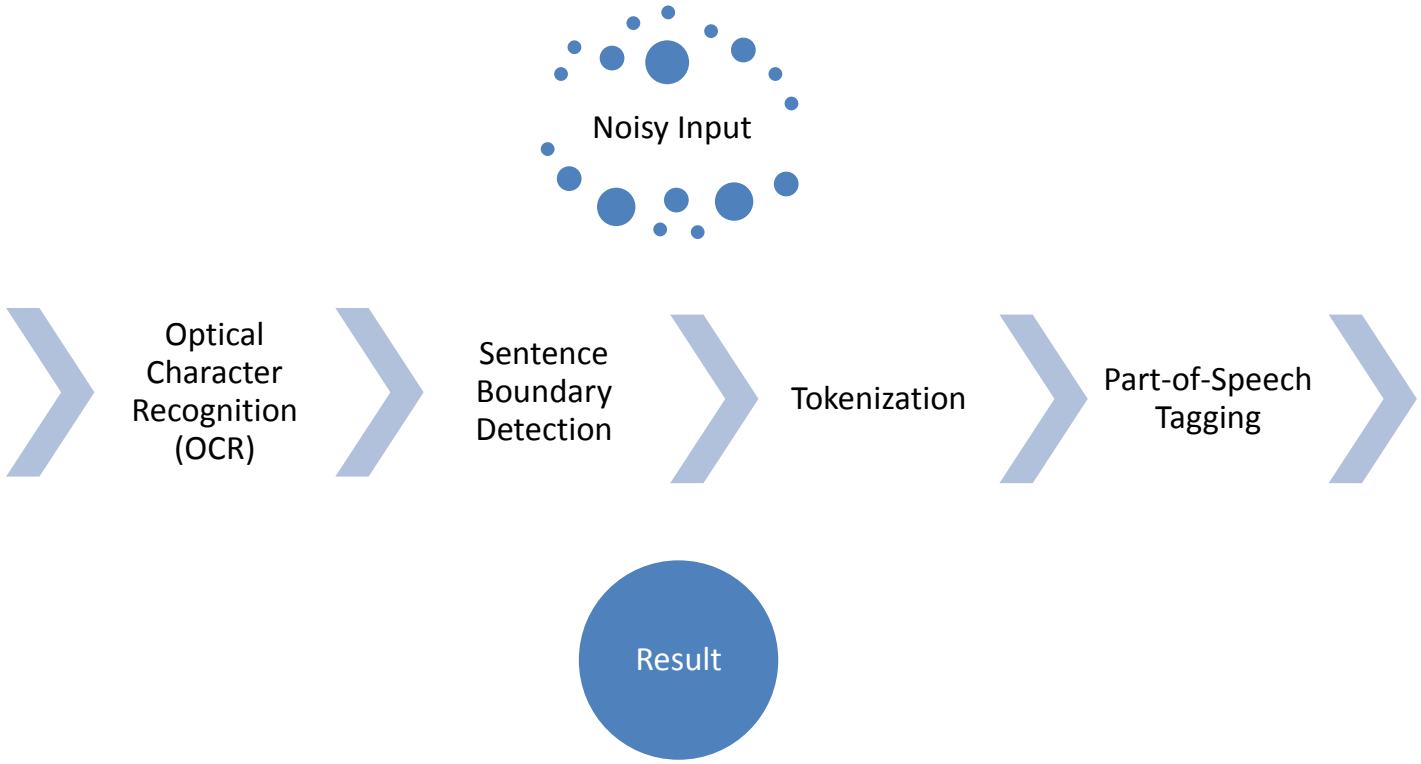
OCR Errors

by Michael Barz

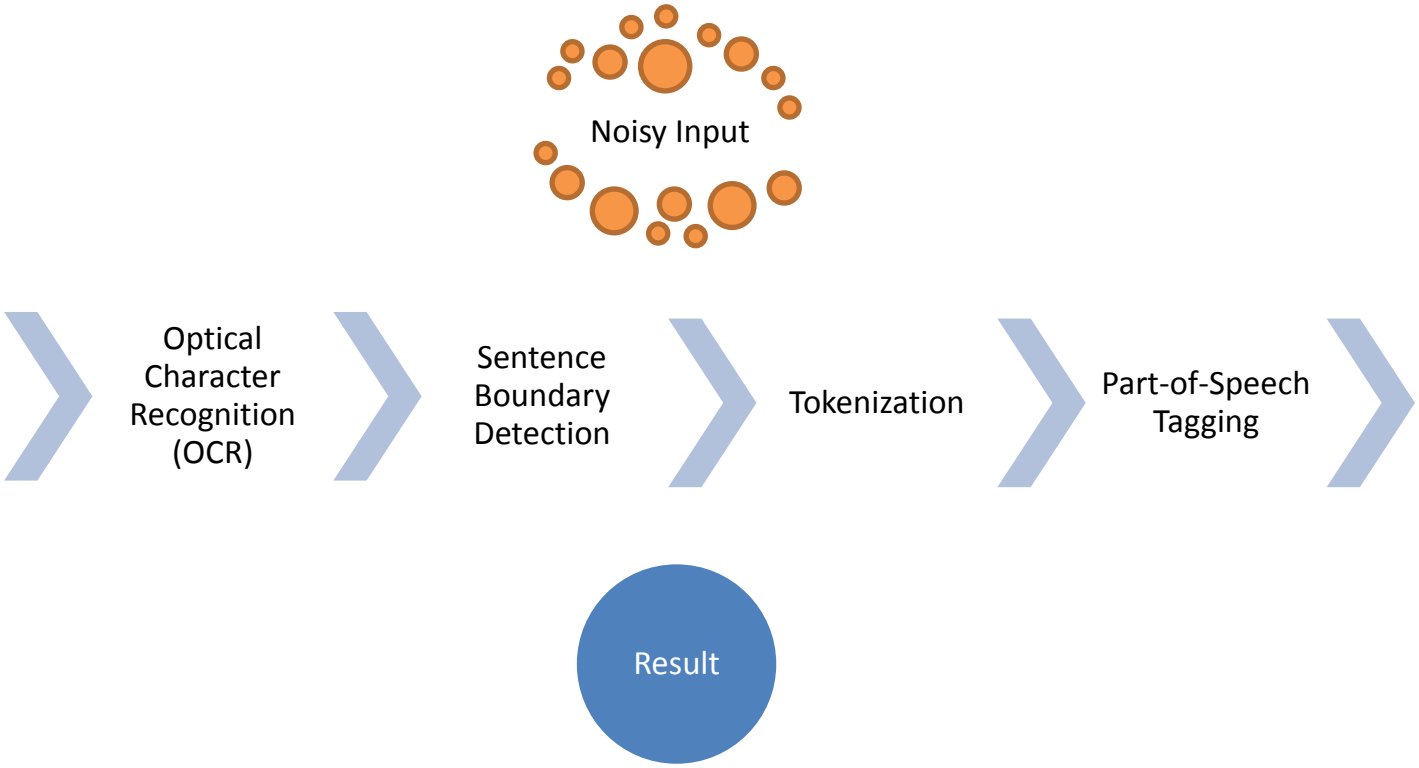
Motivation

- 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 → 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


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

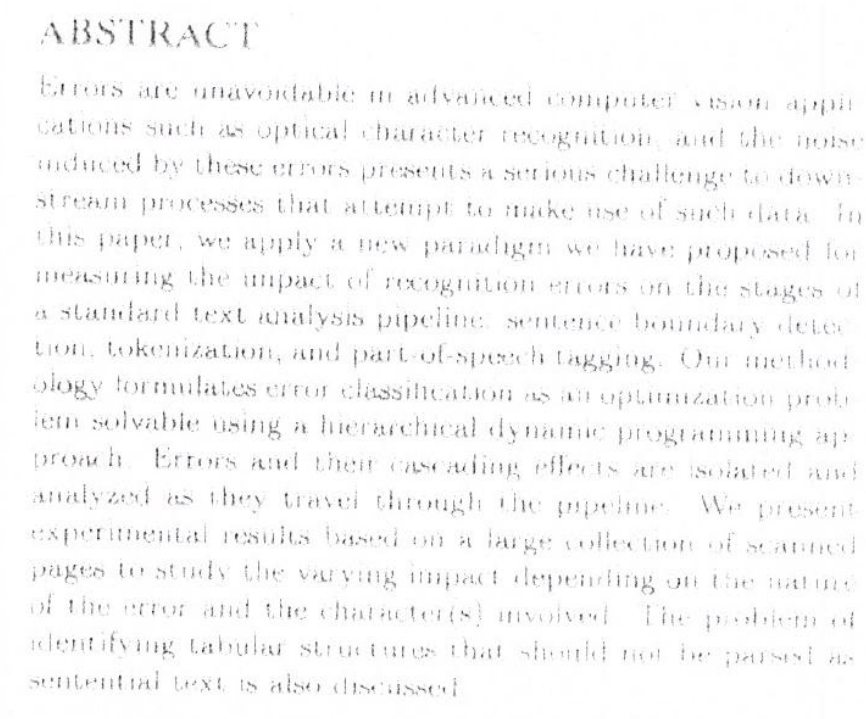
Noisy Input

Clean

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.

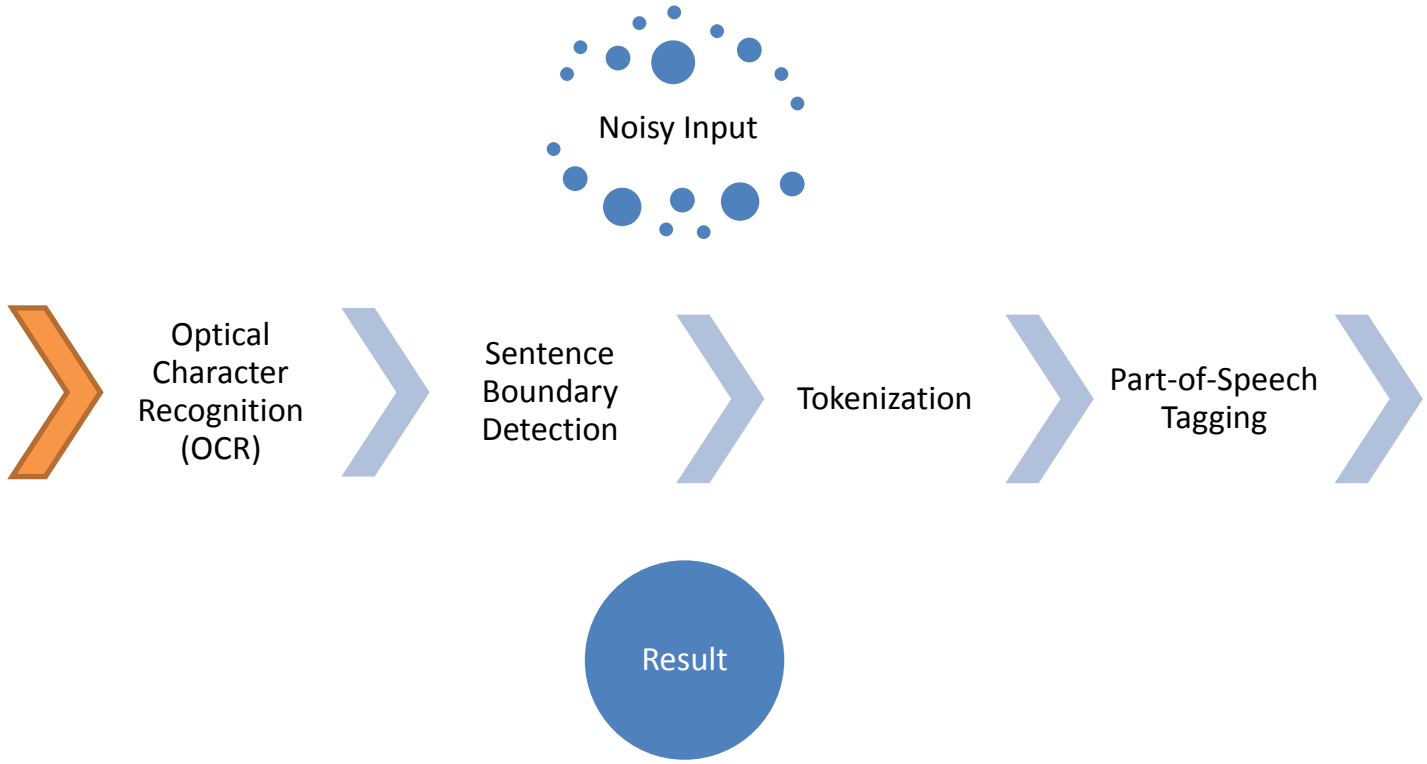
Noisy



Noisy Input

- Generating noisy input to test pipeline
  - Printed digital writing
  - Scanned directly for clean input
  - Repeated copies combined with fax
    → noisy input

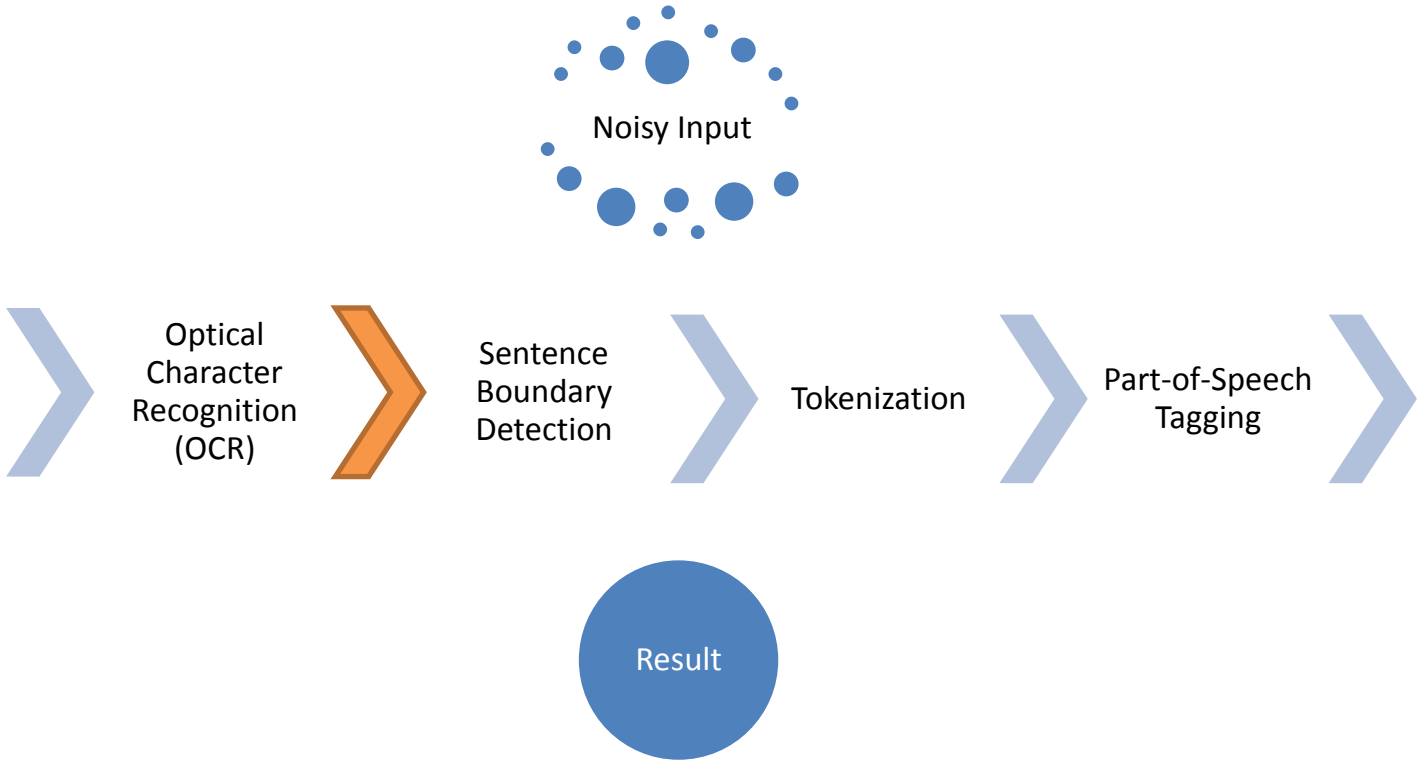
Optical Character Recognition



Optical Character Recognition

- “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

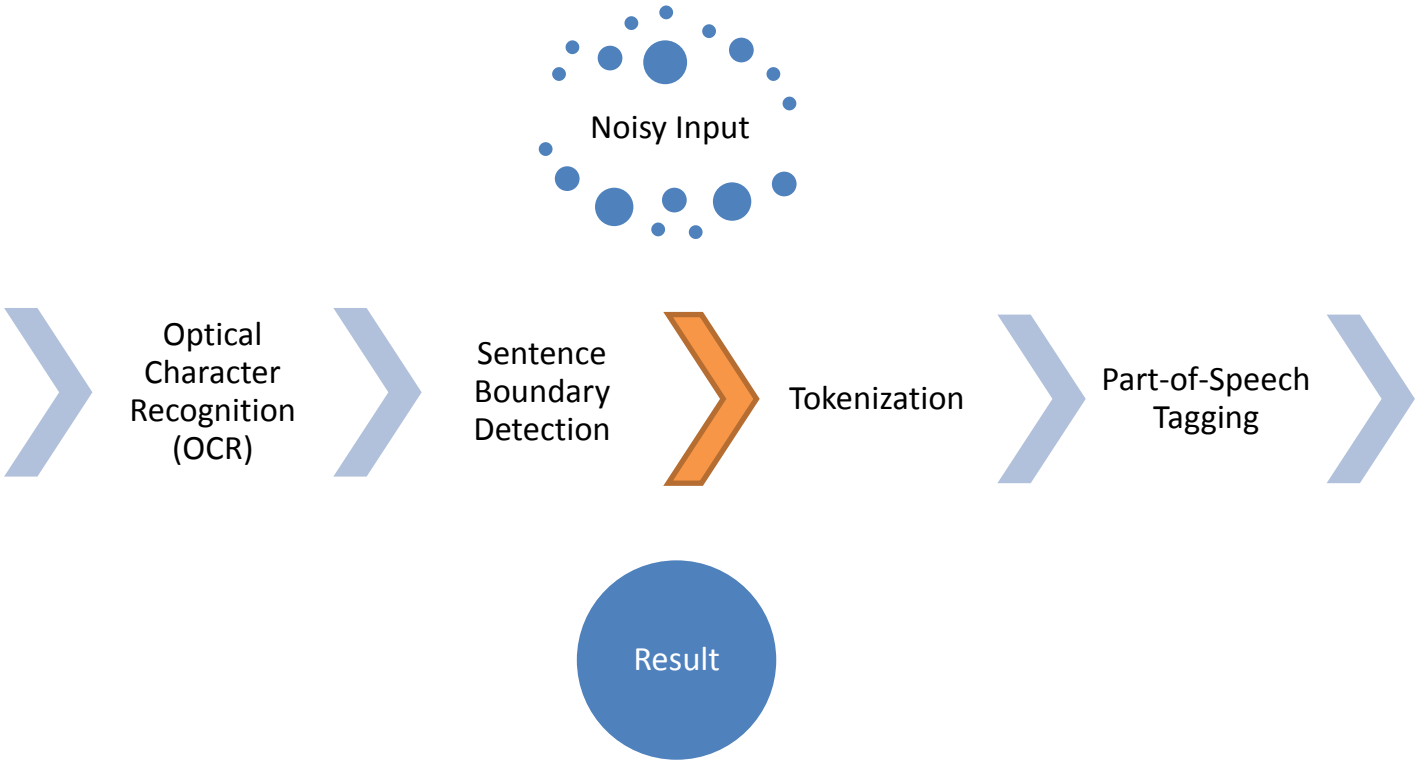
Sentence Boundary Detection



Sentence Boundary Detection

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

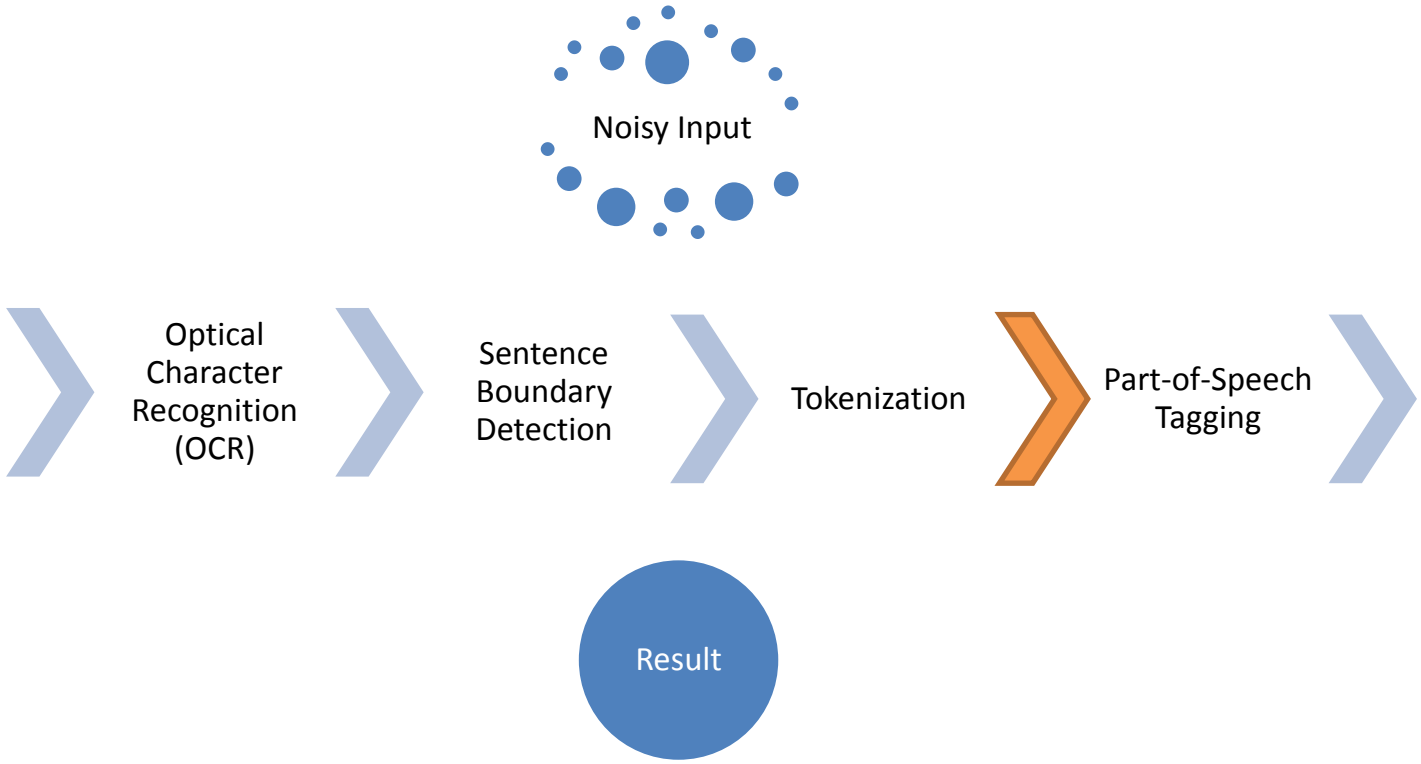
Tokenization



Tokenization

- “breaks it into individual tokens which are delimited by whitespace”
  - Tokens: words, punctuation symbols
- Tool: Penn Treebank tokenizer

Part-of-Speech Tagging



Part-of-Speech Tagging

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

Sample Result

VB	IN	DT	NNS	IN	NNS	RB	.
Look	at	the	crowds	of	water-gazers	there	.

Ground-Truth

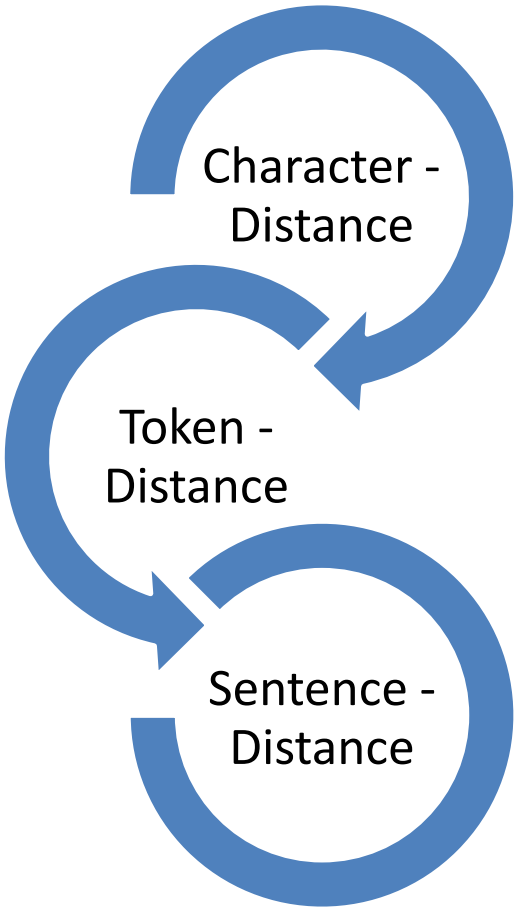
OCR Output

NN	IN	DT	. NNS	IN	NNS	RB	.
_oo_	at	the	, rowds	of	water-gazers	th_re	.

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

$$dist1_{i,j} = \min \begin{cases} dist1_{i-1,j} + c1_{del}(s_i) \\ dist1_{i,j-1} + c1_{ins}(t_j) \\ dist1_{i-1,j-1} + c1_{sub}(s_i, t_j) \end{cases}$$

	ε	**T**	**o**	**r**
ε	0	1	2	3
T	1	0	1	2
i	2	1	1	2
e	3	2	2	2
r	4	3	3	**2**

Performance Evaluation

- Extention: Substitution of more than one sign

$$dist1_{i,j} = \min \begin{cases} dist1_{i-1,j} + c1_{del}(s_i) \\ dist1_{i,j-1} + c1_{ins}(t_j) \\ \min_{1 \le k' \le k,\ 1 \le l' \le l} [\ dist1_{i-k',j-l'} + \\ \qquad c1_{sub_{k:l}}(s_{i-k'+1 \ldots i}, t_{j-l'+1 \ldots j})] \end{cases}$$

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

Evaluation 2005

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

	All Symbols			Punctuation			Whitespace		
	Prec.	Recall	Overall	Prec.	Recall	Overall	Prec.	Recall	Overall
Clean	0.982	0.988	0.988	0.843	0.973	0.912	0.974	0.996	0.985
Light	0.646	0.747	0.790	0.193	0.648	0.315	0.589	0.965	0.730
Dark	0.411	0.575	0.628	0.090	0.686	0.170	0.391	0.884	0.539
Fax	0.584	0.644	0.732	0.201	0.625	0.306	0.608	0.890	0.717

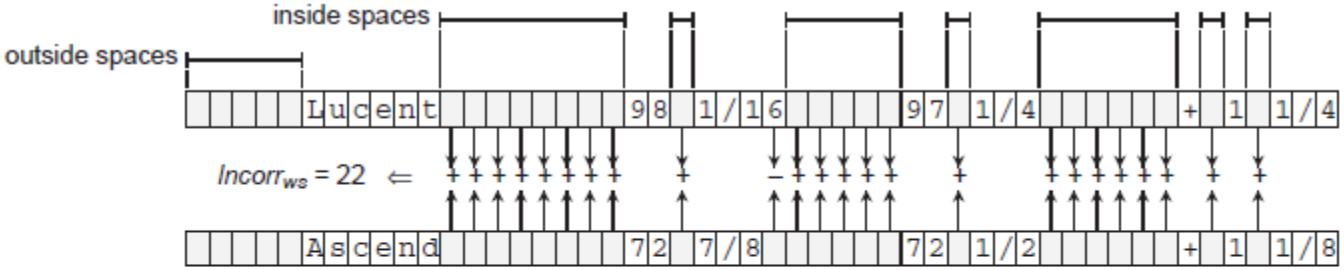
Table 2: Average text processing performance relative to ground-truth.

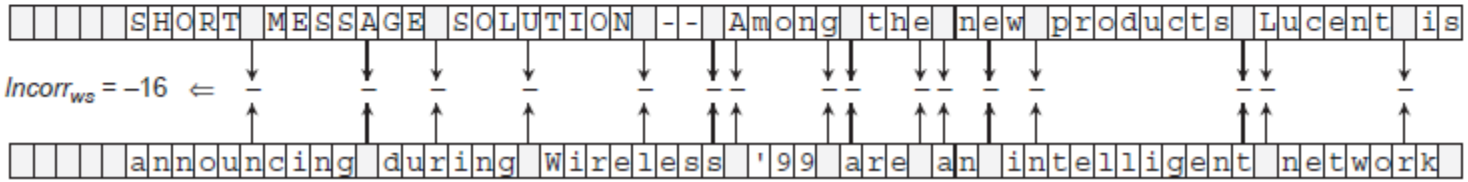
	Sentence Boundaries			Tokenization			Part-of-Speech Tagging		
	Prec.	Recall	Overall	Prec.	Recall	Overall	Prec.	Recall	Overall
Clean	0.939	0.985	0.961	0.975	0.994	0.984	0.953	0.975	0.964
Light	0.648	0.906	0.731	0.646	0.877	0.733	0.307	0.500	0.380
Dark	0.321	0.995	0.405	0.388	0.691	0.479	0.097	0.210	0.132
Fax	0.442	0.987	0.536	0.494	0.674	0.563	0.203	0.303	0.242

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





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

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

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

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

$$tab[i,j] = \max \begin{cases} merit_{pre}(i,[i+1,j]) \; + \; tab[i+1,j] \\ tab[i,j-1] \; + \; merit_{app}([i,j-1],j) \end{cases}$$

Table Spotting

$$tab[i,j] = \max \begin{cases} merit_{pre}(i,[i+1,j]) \; + \; tab[i+1,j] \\ tab[i,j-1] \; + \; merit_{app}([i,j-1],j) \end{cases}$$

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

Evaluation 2008

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

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

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

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

Error identification

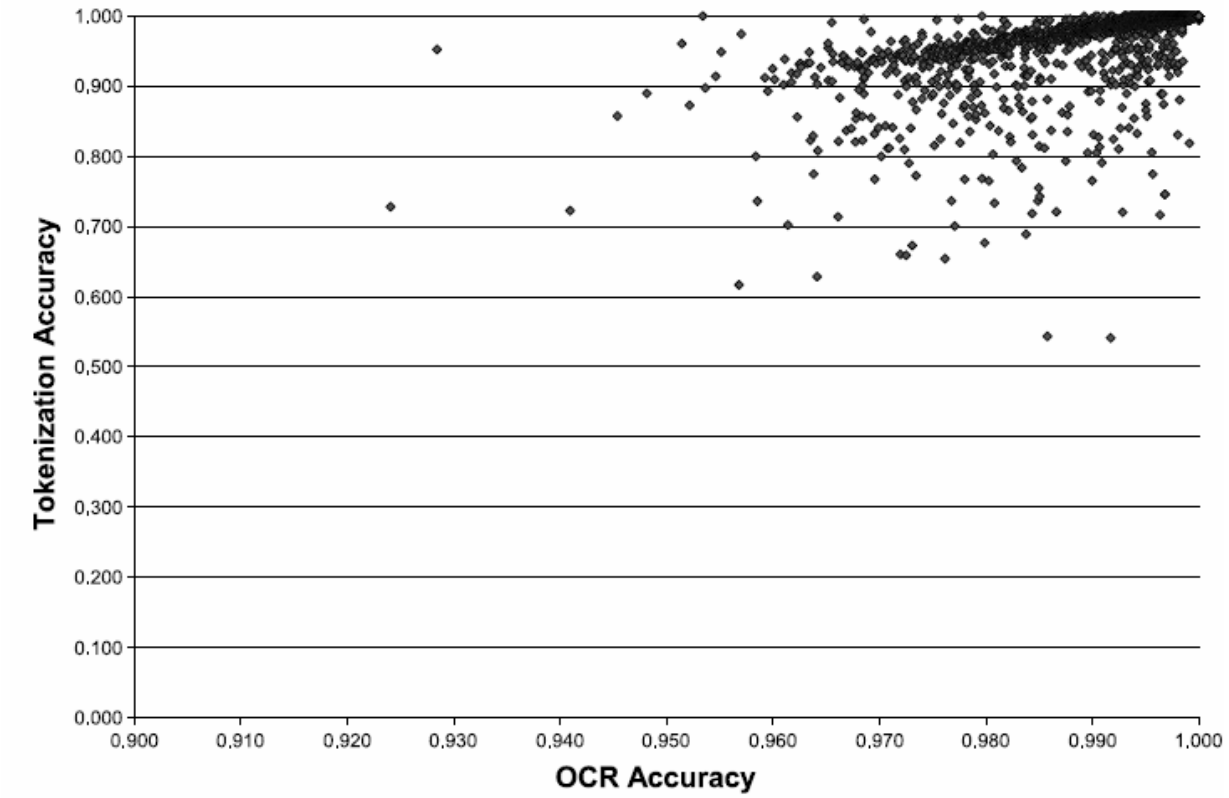


Figure 8: Tokenization accuracy as a function of OCR accuracy.

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!