Document and Content Analysis

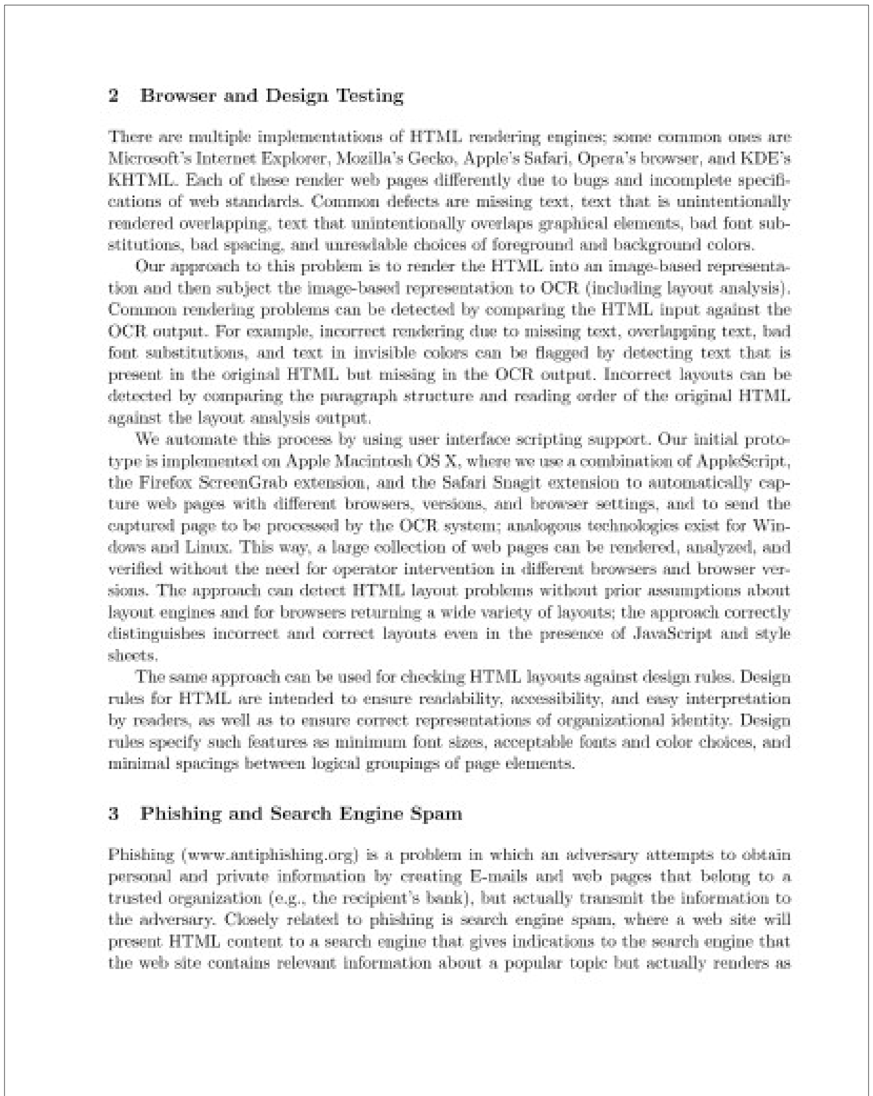
Summer 2009

Lecture 8
Language Modeling

Thomas Breuel
Faisal Shafait

OCR errors

commercial OCR – clean input



2 Browser and Design Testing

There are multiple implementations of HTML rendering engines; some common ones are Microsoft's Internet Explorer, Mozilla's Gecko, Apple's Safari, Opera's browser, and KDE's KHTML. Each of these render web pages differently due to bugs and incomplete specifications of web standards. Common defects are missing text, text that is unintentionally rendered overlapping, text that unintentionally overlaps graphical elements, bad font substitutions, bad spacing, and unreadable choices of foreground and background colors.

Our approach to this problem is to render the HTML into an image-based representation and then subject the image-based representation to OCR (including layout analysis). Common rendering problems can be detected by comparing the HTML input against the OCR output. For example, incorrect rendering due to missing text, overlapping text, bad font substitutions, and text in invisible colors can be flagged by detecting text that is present in the original HTML but missing in the OCR output. Incorrect layouts can be detected by comparing the paragraph structure and reading order of the original HTML against the layout analysis output.

We automate this process by using user interface scripting support. Our initial prototype is implemented on Apple Macintosh OS X, where we use a combination of AppleScript, the Firefox ScreenGrab extension, and the Safari Snagit extension to automatically capture web pages with different browsers, versions, and browser settings, and to send the captured page to be processed by the OCR system; analogous technologies exist for Windows and Linux. This way, a large collection of web pages can be rendered, analyzed, and verified without the need for operator intervention in different browsers and browser versions. The approach can detect HTML layout problems without prior assumptions about layout engines and for browsers returning a wide variety of layouts; the approach correctly distinguishes incorrect and correct layouts even in the presence of JavaScript and style sheets.

The same approach can be used for checking HTML layouts against design rules. Design rules for HTML are intended to ensure readability, accessibility, and easy interpretation by readers, as well as to ensure correct representations of organizational identity. Design rules specify such features as minimum font sizes, acceptable fonts and color choices, and minimal spacings between logical groupings of page elements.

3 Phishing and Search Engine Spam

Phishing (www.antiphishing.org) is a problem in which an adversary attempts to obtain personal and private information by creating E-mails and web pages that belong to a trusted organization (e.g., the recipient's bank), but actually transmit the information to the adversary. Closely related to phishing is search engine spam, where a web site will present HTML content to a search engine that gives indications to the search engine that the web site contains relevant information about a popular topic but actually renders as

2 Browser and Design Testing

There are multiple implementations of HTML rendering engines; some common ones are Microsoft's Internet Explorer, Mozilla's Gecko, Apple's Safari, Opera's browser, and KDE's KHTML. Each of these render web pages differently due to bugs and incomplete specifi- cations of web standards. Common defects are missing text, text that is unintentionally rendered overlapping, text that unintentionally overlaps graphical elements, bad font sub- stitutions, bad spacing, and unreadable choices of foreground and background colors.

Our approach to this problem is to render the HTML into an image-based representa- tion and then subject the image-based representation to OCR (including layout analysis)...

commercial OCR – scientific publications

Indeed, it follows from (3.5′) in view of (4.39) that

$$
\begin{aligned}
(4.40)\qquad \mathcal{E}_n^{-1}(U)\,X_n^{\pi_\alpha} &= (1-\alpha)\mathbb{C} + \sum_{k=1}^{n} \mathcal{E}_k^{-1}(U)\,\gamma_k^{\alpha}\,S_{k-1}\,(\rho_k - r_k) \\
&= (1-\alpha)\mathbb{C} + \sum_{k=1}^{n} \mathcal{E}_k^{-1}(U)\,S_{k-1}\,\gamma_k^{*}\,(\rho_k - r_k) \\
&\quad - \sum_{k=1}^{n} \mathcal{E}_k^{-1}(U)\,S_{k-1}\,\varphi_k\,\mathbb{C}\,(\rho_k - r_k) \\
&= (1-\alpha)\mathbb{C} - (1-\alpha)\,\mathbb{C} + M_n^{\mathbb{C}} - \mathbb{C}\,\mathbf{M}_n^{\alpha}.
\end{aligned}
$$

From (4.40),

$$\mathcal{E}_N^{-1}(U)\,X_N^{\pi_\alpha} = \mathcal{E}_N^{-1}(U)\,f - \mathbb{C}\,\mathbf{I}_{\{Z_N<\lambda\}}$$

and hence

$$\mathcal{E}_N^{-1}(U)\,X_N^{\pi_\alpha} \geq \mathcal{E}_N^{-1}(U)\,f - \mathbb{C}.$$

The last inequality means that $\pi_\alpha \in \mathrm{SF}(f, N)$.

Further, it follows from (4.38) that

$$
\begin{aligned}
(4.41)\qquad \mathbf{P}^*\{X_N^{\pi_\alpha} \geq f\} &= \mathbf{P}^*\{f - \mathbb{C}\,\mathcal{E}_N(U)\,\mathbf{I}_{\{Z_N<\lambda\}} \geq f\} \\
&= \mathbf{P}^*\{\mathbf{I}_{\{Z_N<\lambda\}} \leq 0\} = \mathbf{P}^*\{Z_N \geq \lambda\} = (1-\alpha).
\end{aligned}
$$

Finally, we get from (4.38) and (4.41) that

$$
\begin{aligned}
(4.42)\qquad \mathbf{P}\,\{X_N^{\pi_\alpha} \geq f\} &= \mathbf{E}^*\,\mathbf{I}_{\{X_N^{\pi_\alpha}\geq f\}}Z_N \\
&\geq \mathbf{E}^*\,\mathbf{I}_{\{X_N^{\pi_\alpha}\geq f\}}\mathbf{I}_{\{Z_N\geq\lambda\}}Z_N \\
&\geq \lambda\,(1-\alpha) \geq 1-\alpha.
\end{aligned}
$$

The relations (4.41) and (4.42) show that the condition (4.35) holds for the strategy π_α, and hence π_α is an α-$((1-\alpha)\mathbb{C}, f, N)$-hedge.

What has been obtained shows that it is possible to hedge a contingent claim *with a specified probability* $(1-\alpha)$. Further, *the initial funds can be reduced by the amount* $\alpha\,\mathbb{C}$, though with a *risk* α the accepted contingent claim cannot be repaid.

Problems

4.1. Prove that on a no-arbitrage (B, S)-market we have for a standard European option to buy (sell) that $\mathbb{C}(N_2) \geq \mathbb{C}(N_1)$ (respectively, $\mathbb{P}(N_2) \geq \mathbb{P}(N_1)$) when $N_2 \geq N_1$.

4.2. Prove that the fair price $\mathbb{C} = \mathbb{C}(N, S_0, K)$ of a standard European option to buy, where N is the exercise time, S_0 is the initial price of a share, and K is the exercise price, has the following properties:

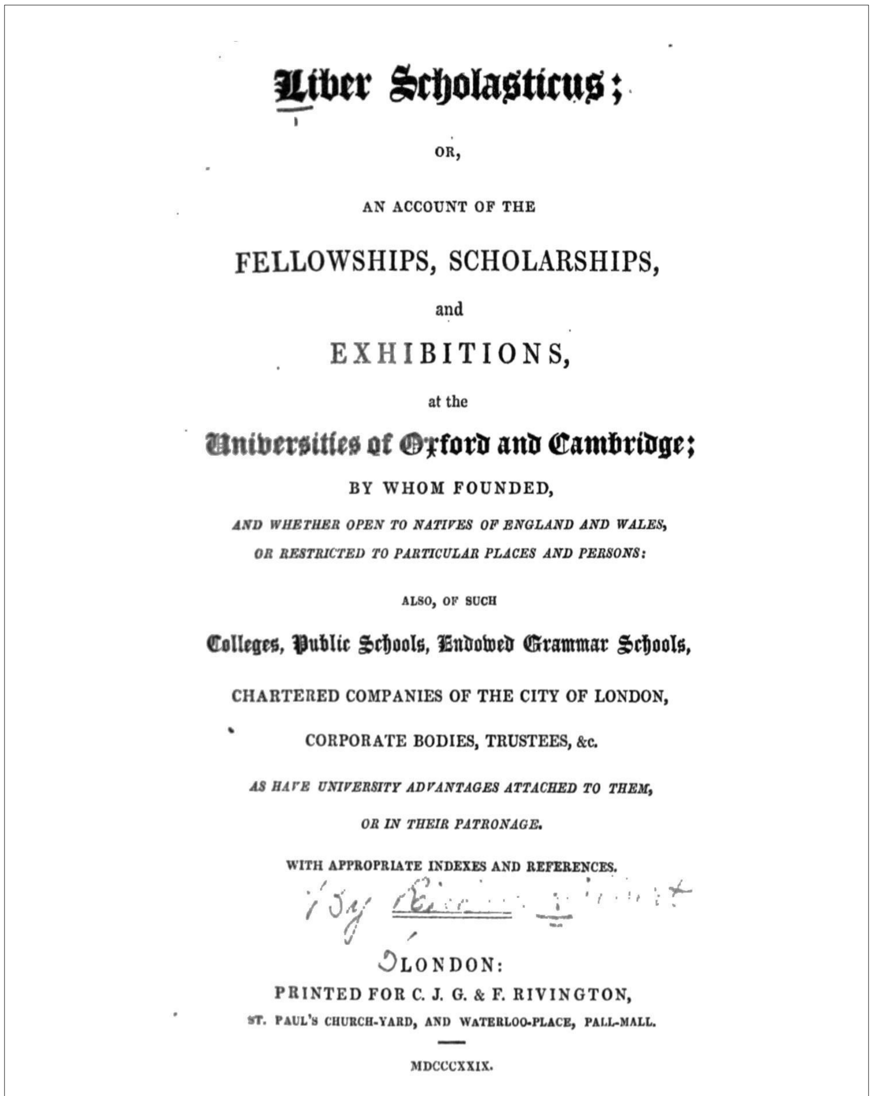
a) $\mathbb{C}(S_0, K)$ is monotone in S_0 and K;

d) $\mathbb{C}(S_0, K)$ is convex in S_0 and K;

c) $\mathbb{C}(\lambda S_0, \lambda K) = \lambda\,\mathbb{C}(S_0, K)$ for $\lambda > 0$.

Indeed, it follows from (3.5') in view of (4.39) that (4.40) S k=l fc=i = (1 - a)C - (1 - a) C + From (4.40), �^(u)x^=� and hence The last inequality means that TTQ 6 SF(f,N). Further, it follows from (4.38) that (4.41) P'{X# > /} = P*{/ -C�N(U) I{ZN f} = P'{I{ZN< 0} = P*{ZN > A} = (1 - a). Finally, we get from (4.38) and (4.41) that (4.42) P{X^>/}=E'IW� > A(l-a) > 1-a. The relations (4.41) and (4.42) show that the condition (4.35) holds for the strategy na, and hence TTQ is an a-((l - a)C, /, A^)-hedge. What has been obtained shows that it is possible to hedge a contingent claim with a specified probability (1 � a). Further, the initial funds can be reduced by the amount a C, though with a risk a the accepted contingent claim cannot be repaid. PROBLEMS 4.1. Prove that on a no-arbitrage (B, 5)-market we have for a standard Euro- pean option to buy (sell) that C(7V2) > C(Ni) (respectively, P(JV2) > P(M)) when 4.2. Prove that the fair price C = C(N, So, K) of a standard European option to buy, where N is the exercise time, So is the initial price of a share, and K is the exercise price, has the following properties: a) C(S0, K) is monotone in So and K; d) C(So, K) is convex in So

commercial OCR – unusual fonts



Liber Scholasticus;

OR,

AN ACCOUNT OF THE

FELLOWSHIPS, SCHOLARSHIPS,

and

EXHIBITIONS,

at the

Universities of Oxford and Cambridge;

BY WHOM FOUNDED,

AND WHETHER OPEN TO NATIVES OF ENGLAND AND WALES,
OR RESTRICTED TO PARTICULAR PLACES AND PERSONS:

ALSO, OF SUCH

Colleges, Public Schools, Endowed Grammar Schools,

CHARTERED COMPANIES OF THE CITY OF LONDON,

CORPORATE BODIES, TRUSTEES, &c.

AS HAVE UNIVERSITY ADVANTAGES ATTACHED TO THEM,
OR IN THEIR PATRONAGE.

WITH APPROPRIATE INDEXES AND REFERENCES.

LONDON:
PRINTED FOR C. J. G. & F. RIVINGTON,
ST. PAUL'S CHURCH-YARD, AND WATERLOO-PLACE, PALL-MALL.

MDCCCXXIX.

OR,
AN ACCOUNT OP THE
FELLOWSHIPS, SCHOLARSHIPS,
and
EXHIBITIONS,
at the
attttonvitto of <C2><A9>Tforfc anft
<E2><82><AC>amfitiU0
BY WHOM FOUNDED,
J.VJ> UHKrilKK OPEff TO IfATIfES OF
SNOLAND AND WALES,
Ott RKiTRICTEU TO PARTICULAR
PLACES AND PERSONS;
ALSO, OF SUCH
CoKrgrs, IJutlir $rf)ool6, Kniutotti
(Grammar 5rf)ool
CHABTERED COMPANIES OF THE CITY
OF LONDON,
CORPORATE BODIES, TRUSTEES, &c.
At BArS OXIrESSJTY ADrANTAOES
ATTACHED TO TBEX,
OS IN THEM PATRONAGE.
WITH APPROPRIATE INDEXES AND
REFERENCES.
^LONDON:
PRINTED FOR C. J. O. & F. RIV1NGTON,
.   PAI L's CIlf RCII.YAlin, AMI
WATERLOO.PLACE,   PALUMALL.

MDCCCXXIX.

commercial OCR – languages

PRÓLOGO.

Voy á leerte unos manuscritos, que mas desvelos costó á mi padre el sustraerlos á tu curiosidad, que el escribirlos. Sé que cometo una imprudencia satisfaciendo un femenil deseo que te acarreará muchos dolores; pero contigo mas quiero pecar de tolerante que de severo. Profanaré con el secreto la memoria de mi buen padre, mas añadiré quilates á tu cariño: entre los respetos debidos á la memoria de un padre muerto, y el amor

â¬*Mlnv-
Toy aleertennof n^m^ritn. qaeva* desTdos eosto
4 mi padre d snstnerlos a tu oniosidad, qae d eseri-
birlos. Se" qne cometa ana impradtncia iilirfirirÂ«dn on
femenil deseo qne te aearreara modiM dokns; pcro ew-
tigo mas quiero pecar de tolerant* qne de wrcro. Pra-
fanart COD el secrete la memoria de mi boen padre.
mas anadirt qoilates a tu carioo: eatre 1Â« respeto* de-
bidos a. la memoria de on padre nmerlo, j d amor

measuring OCR accuracy

- identify the better OCR system
- guide development & improvements
- monitor production processes
- charge penalties during production

OCR errors = typos?

kinds+sources of OCR errors

- **preprocessing**
  - thresholding, page frame detection, ...
- **layout analysis**
  - block detection, text/image segmentation, line finding, ...
- **character segmentation**
  - touching characters, split characters
- **character recognition**
  - shape confusions, unknown chars, new fonts, ...
- **language modeling**
  - out of dictionary words, ...

character-level errors

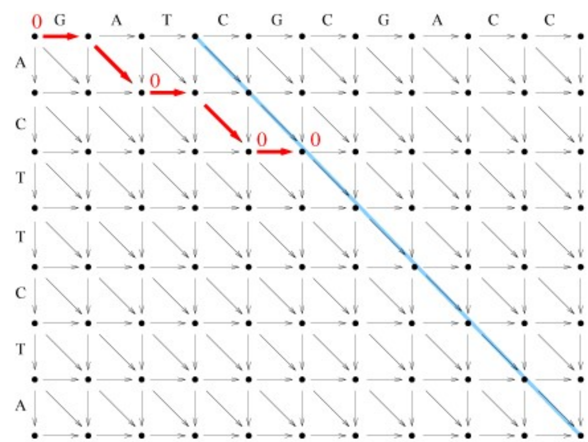
- **confusions**
  - character shape confusions
- **insertions**
  - high threshold → noise
  - non-text elements
  - low threshold → split characters
  - confusable characters
- **deletions**
  - low threshold → missing characters
  - high threshold → touching characters
  - page-elements touching characters

OCR errors = typos + much more

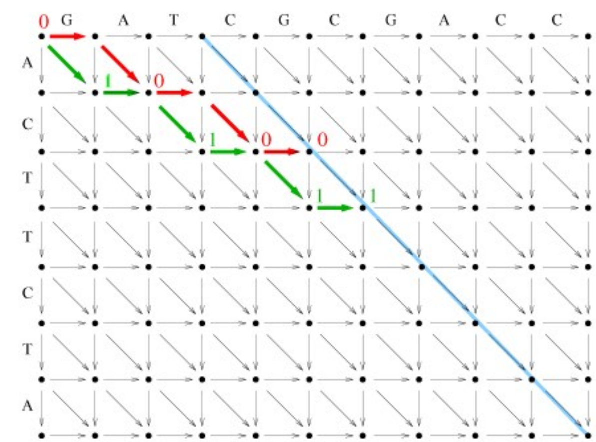
string edit distance

- **“number of corrections necessary”**
- **motivated by manual correction**
- **assign costs to…**
  - changing a character
  - inserting a character
  - deleting a character
- **implement using dynamic programming alg.**

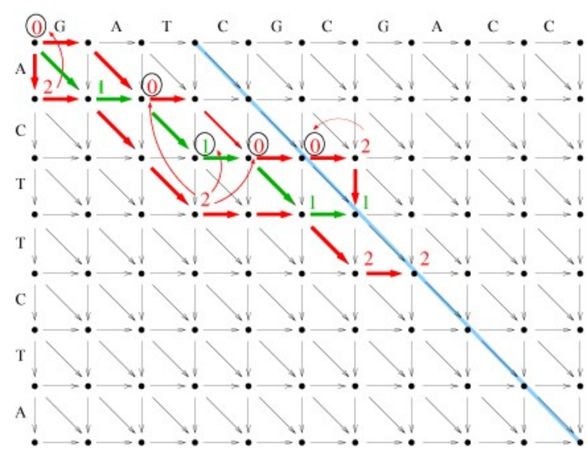
string edit distance



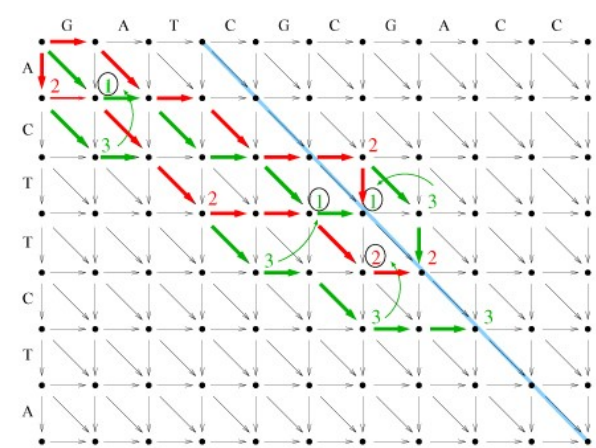
a. Score 0 iteration



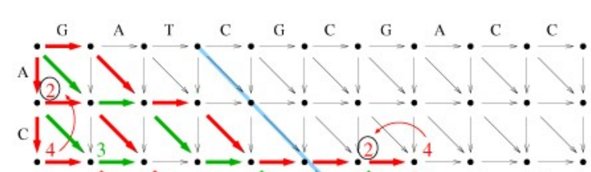
a. Score 1 iteration



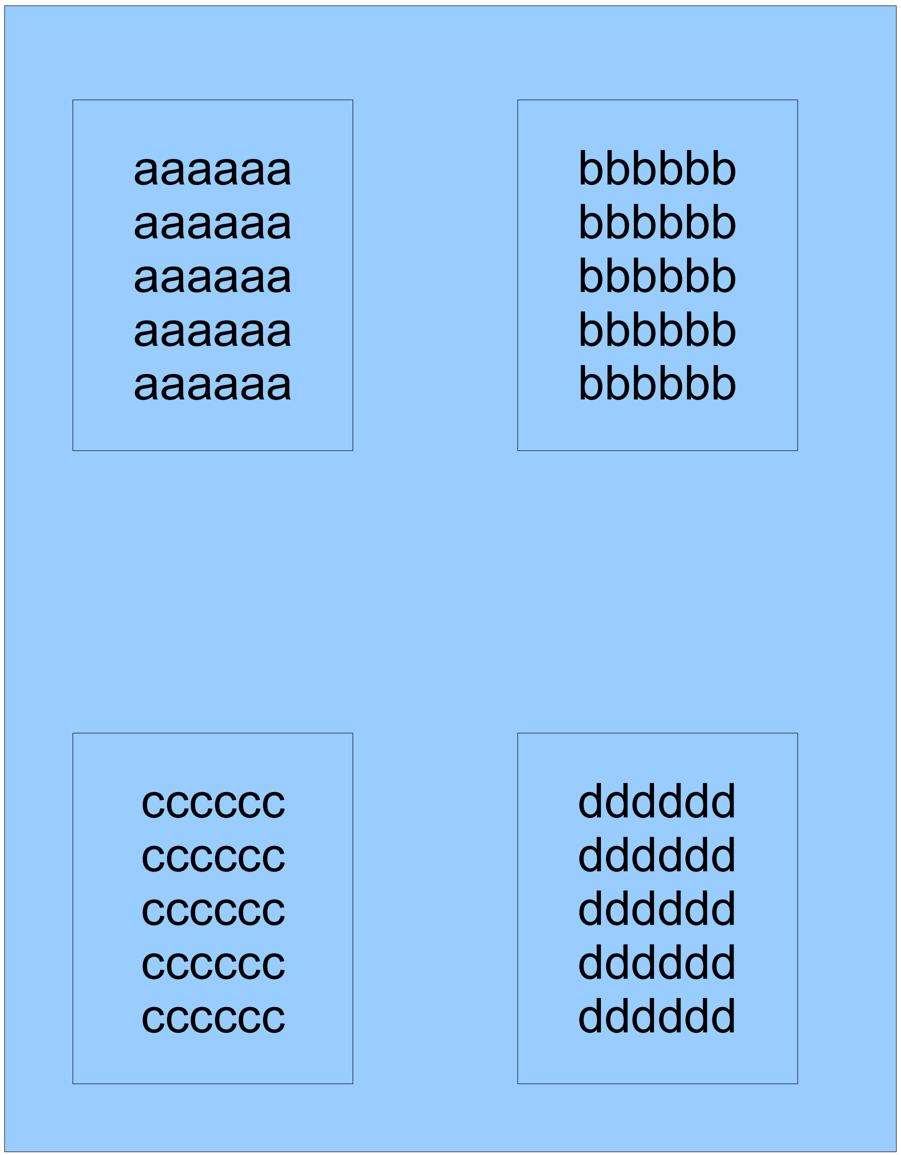
a. Score 2 iteration



a. Score 3 iteration



layout errors?



correct

aaaaaa aaaaaa ... bbbbbb
bbbbbb ... cccccc cccccc ...
dddddd dddddd ...

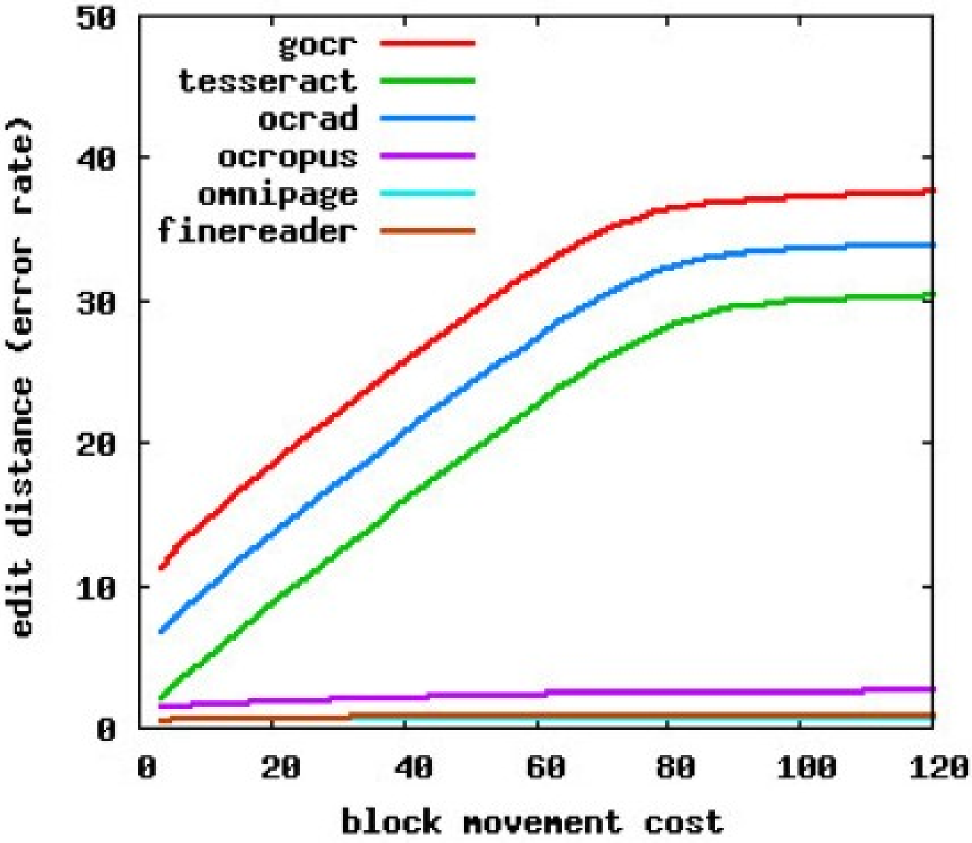
actual

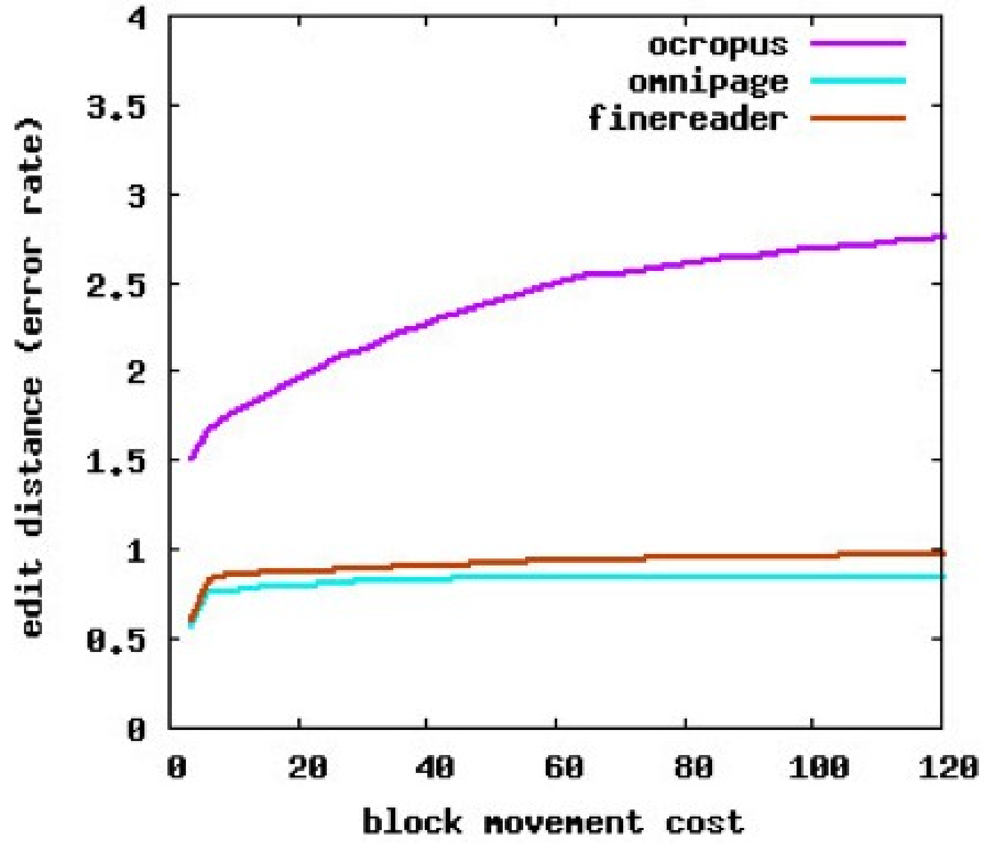
aaaaaa aaaaaa ... cccccc
cccccc ... bbbbbb bbbbbb ...
dddddd dddddd ...

string edit with block move

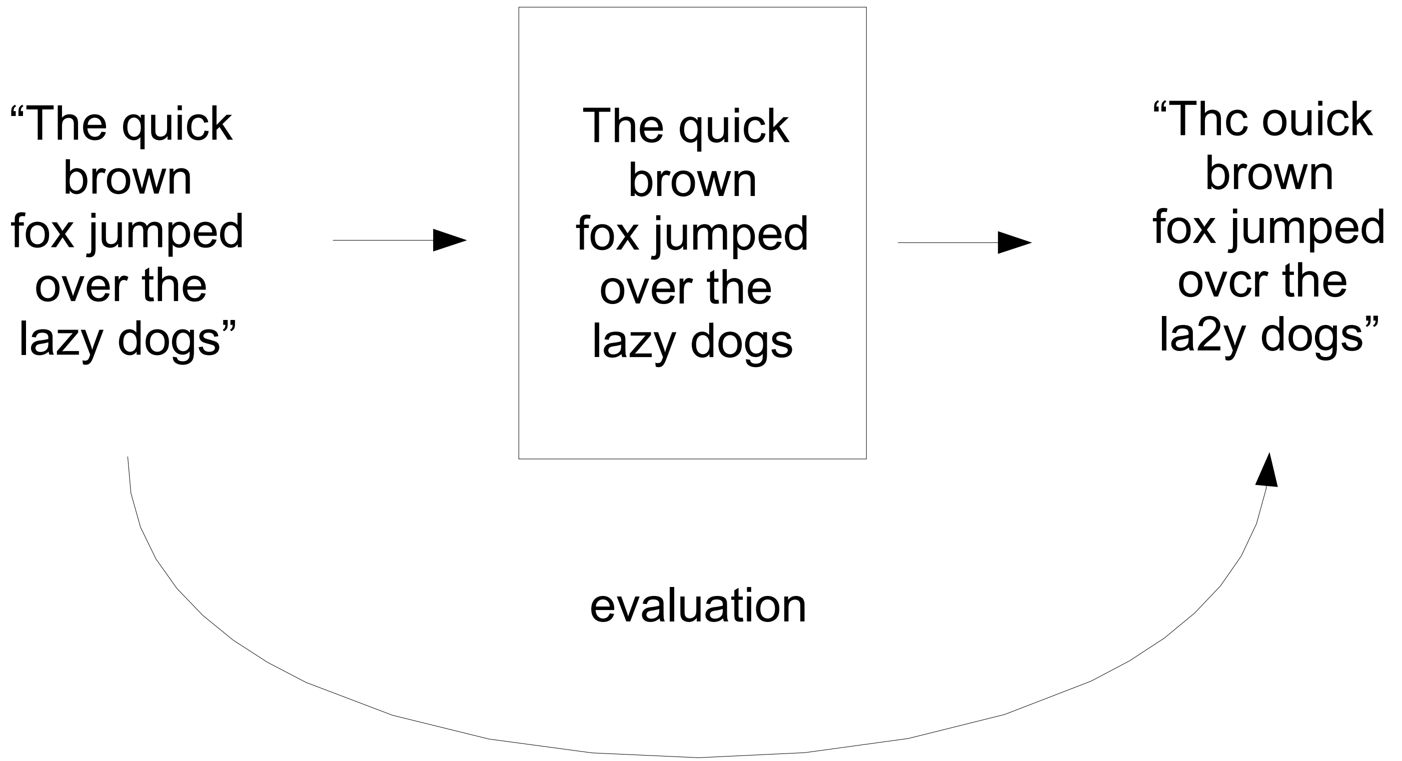
- **basic operations**
  - insert character
  - delete character
  - change character
  - move a block of characters (cut+paste)
- **block move cost is a parameter**

OCR evaluation w/block move cost

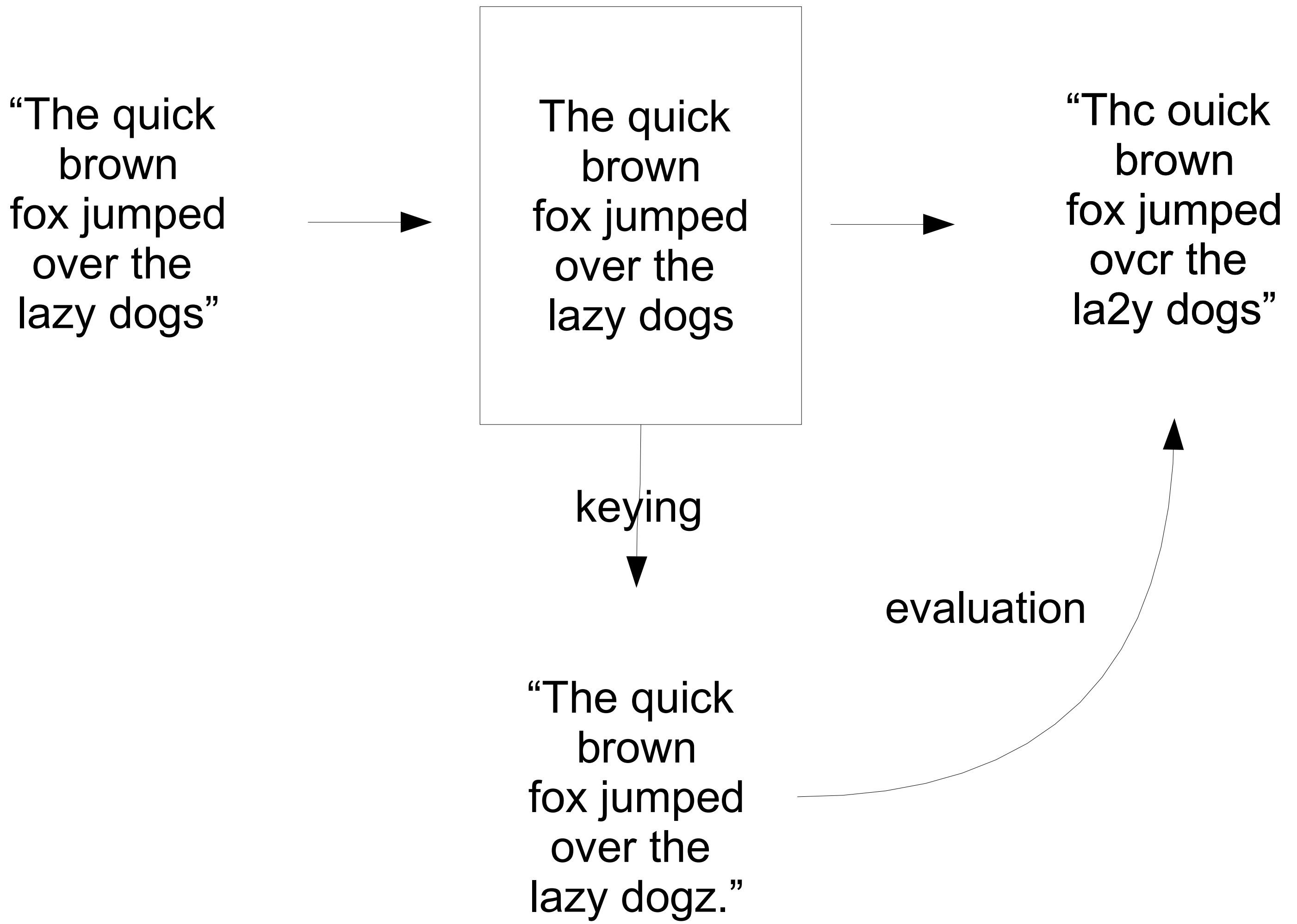




ground truth



manual keying



ground truth

- **true ground truth**
  - source document
- **usual ground truth**
  - manual keying
    - error rates comparable to OCR
  - sources of errors
    - typos, language bias (e.g., names), ...
  - solution
    - double keying with reconciliation
    - triple keying with reconciliation
    - typists who don't know the language

double keying, triple keying?

- what's the actual error rate?
- statistical correlations between typists?

layout evaluation

- **indirect layout evaluation**
  - edit distance with block move
- **direct layout evaluation**
  - compare geometric partition of documents

approaches to correction

spell checking

- **simple idea...**
  - people mistype... use spell correction
  - OCR system mistype... use spell correction

spell correction

- **divide the source text into words**
- **for each word, look it up in the dictionary**
- **if not found...**
    - find best matching word(s) by edit distance
    - if unique, replace
    - if not unique, resolve ambiguity somehow

spell correction issues

- **kinds of errors**
  - typos...
    - characters close to each other on the keyboard
    - phonetic mistakes
    - common patterns
  - OCR
    - segmentation errors “cl”/“d”, “rn”/“m”
    - shape confusions “e”/“c”, “2”/“Z”
    - noise “.”/
  - manual spell correction systems may not be suitable
- **making it fast**

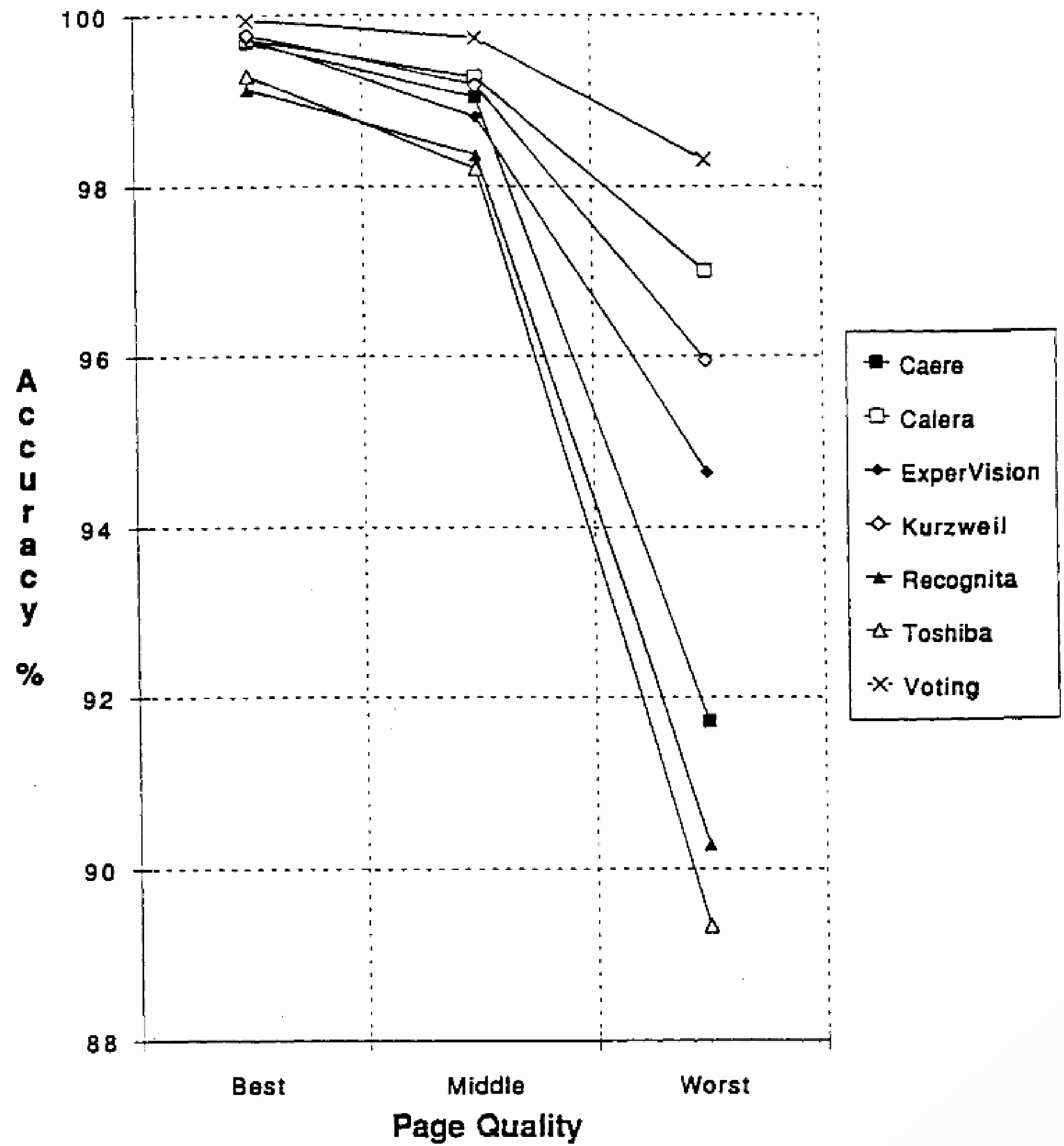
OCR voting

OCR voting

- **simple idea**
  - run 3 or more OCR systems
  - find corresponding words in the outputs
  - pick the word that's most frequently voted for
  - break ties somehow

	# Errors	% Accuracy
Caere OmniPage Professional	8841	96.83
Calera RS 9000	3709	98.67
ExperVision TypeReader	6318	97.73
Kurzweil 5200	4716	98.31
Recognita Plus	11282	95.95
Toshiba ExpressReader	12169	95.64
ISRI Voting Algorithm	1867	99.33

Table 1. Accuracy Statistics for the Entire Sample



100
98
96
94
92
90
88
Accuracy %
Best
Middle
Worst
Page Quality
Caere
Calera
ExperVision
Kurzweil
Recognita
Toshiba
Voting

issues with OCR voting

- how do we cope with misaligned outputs?
- should some systems have precedence?
- is the extra processing time worth it?
- is the output the best place to combine?
- what if I add the same OCR system twice?
- character level or word level?

OCR voting statistically

- **each system tries to compute $\arg\max_{word} P(word \mid image)$**
- **how do we combine $word_1$, $word_2$, $word_3$?**
- **confidence scores**
    - return word + $f(P(word \mid image))$
    - f is usually monotonic (larger for higher posterior probability)
    - f may depend on the characters in the word
    - how do we compare/combine?

fully statistical OCR

OCR steps

- cut apart pages into text lines and images
- cut apart each text line into characters
- generate segmentation graph
- recognize each character
- find the best path through the graph
- re-assemble the page text

Can we make this probabilistically sound?

Bayes optimal solution

- **lowest prob. error = highest posterior prob.**
- **arg max P(string | image)**

per character scores

	t	h	e	q	u	i	c	k	b	r	o	w	n
a	.018	.018	.006	.003	.010	.013	.015	.009	.002	.011	.013	.014	.011
b	.005	.001	.013	.002	.003	.009	.017	.003	.670	.018	.009	.018	.013
c	.014	.010	.019	.015	.016	.015	.798	.010	.014	.014	.016	.018	.004
d	.009	.013	.013	.011	.004	.006	.005	.005	.015	.011	.006	.008	.013
e	.009	.017	.877	.010	.007	.006	.007	.000	.019	.017	.020	.014	.013
f	.015	.004	.008	.009	.001	.005	.013	.010	.019	.006	.001	.003	.012
g	.006	.018	.003	.012	.007	.017	.001	.004	.006	.008	.009	.014	.008
h	.009	.793	.011	.008	.003	.004	.003	.007	.012	.012	.008	.017	.005
i	.018	.016	.012	.019	.019	.778	.004	.006	.018	.007	.018	.004	.004
j	.019	.008	.010	.007	.017	.004	.016	.006	.015	.004	.009	.019	.007
k	.016	.010	.019	.004	.008	.004	.002	.830	.004	.016	.001	.003	.020
l	.006	.000	.007	.004	.004	.010	.003	.012	.000	.011	.009	.004	.007
m	.015	.019	.011	.004	.017	.018	.020	.008	.017	.004	.015	.001	.005
n	.019	.017	.006	.001	.017	.013	.002	.003	.017	.006	.014	.000	.876
o	.018	.014	.011	.007	.018	.018	.002	.016	.009	.006	.779	.014	.007
p	.006	.001	.004	.009	.016	.005	.015	.015	.002	.001	.017	.019	.014
q	.018	.002	.011	.631	.004	.016	.011	.002	.009	.003	.009	.007	.001
r	.011	.004	.010	.017	.017	.017	.002	.018	.001	.530	.012	.015	.006
s	.006	.010	.008	.007	.007	.006	.001	.005	.008	.012	.009	.004	.003
t	.890	.011	.013	.006	.020	.001	.007	.011	.004	.017	.008	.002	.014
u	.010	.000	.015	.002	.678	.015	.001	.008	.005	.009	.015	.012	.015
v	.016	.017	.003	.008	.006	.007	.012	.017	.000	.018	.017	.001	.005
w	.008	.005	.002	.015	.007	.015	.005	.007	.010	.007	.002	.664	.011
x	.015	.007	.019	.010	.019	.014	.006	.016	.018	.014	.003	.005	.006
y	.020	.005	.003	.017	.006	.008	.005	.011	.010	.001	.018	.005	.006
z	.004	.015	.013	.006	.006	.008	.013	.005	.018	.012	.020	.004	.008

arg max $P(c_i| x)$

	t	h	e		q	u	i	c	k		b	r	o	w	n
a	.017	.009	.013		.015	.018	.018	.018	.003		.007	.017	.013	.015	.009
b	.005	.009	.001		.002	.015	.001	.001	.003		**.670**	.014	.001	.014	.007
c	.014	.016	.010		.001	.008	.006	**.798**	.015		.005	.014	.009	.020	.013
d	.006	.018	.016		.014	.015	.008	.005	.009		.004	.013	.018	.005	.015
e	.013	.007	**.877**		.016	.007	.008	.009	.014		.002	.005	.004	.003	.014
f	.009	.018	.019		.004	.007	.019	.016	.013		.017	.012	.007	.012	.000
g	.017	.017	.014		.014	.002	.009	.008	.009		.004	.014	.017	.014	.009
h	.019	**.793**	.019		.012	.002	.013	.002	.000		.011	.005	.007	.010	.001
i	.001	.007	.013		.008	.020	**.778**	.013	.005		.016	.007	.019	.018	.016
j	.007	.007	.020		.001	.004	.014	.010	.003		.012	.002	.005	.005	.004
k	.006	.016	.009		.013	.007	.010	.014	**.830**		.014	.003	.002	.013	.017
l	.007	.009	.004		.006	.007	.019	.013	.014		.019	.014	.018	.013	.004
m	.001	.005	.006		.007	.010	.010	.012	.007		.019	.005	.013	.009	.019
n	.007	.013	.001		.000	.010	.008	.009	.013		.014	.017	.013	.007	**.876**
o	.011	.012	.001		.010	.005	.005	.011	.010		.011	.018	**.779**	.000	.001
p	.010	.007	.000		.015	.000	.010	.014	.007		.002	.016	.007	.012	.004
q	.016	.005	.018		**.631**	.013	.018	.005	.005		.010	.006	.014	.014	.011
r	.005	.004	.002		.003	.005	.003	.013	.011		.004	**.530**	.008	.011	.013
s	.003	.018	.015		.018	.005	.007	.002	.002		.012	.019	.015	.009	.004
t	**.890**	.020	.019		.009	.014	.013	.000	.020		.018	.002	.003	.002	.005
u	.016	.014	.009		.004	**.678**	.005	.002	.007		.003	.017	.005	.009	.004
v	.007	.011	.016		.006	.006	.005	.010	.006		.004	.019	.000	.017	.019
w	.020	.015	.001		.003	.017	.007	.019	.011		.015	.003	.016	**.664**	.017
x	.009	.019	.014		.014	.008	.018	.001	.018		.013	.007	.004	.019	.017
y	.010	.003	.016		.010	.000	.015	.010	.015		.017	.013	.012	.003	.011
z	.003	.018	.014		.019	.016	.004	.018	.010		.018	.006	.007	.019	.004

arg max $P(c_i| x)$

	t	h	e		q	u	i	c	k		b	r	o	w	n
a	.017	.009	.013		.015	.018	.018	.018	.003		.007	.017	.013	.015	.009
b	.005	.009	.001		.002	.015	.001	.001	.003		**.670**	.014	.001	.014	.007
c	.014	.016	.010		.001	.008	.006	**.798**	.015		.005	.014	.009	.020	.013
d	.006	.018	.016		.014	.015	.008	.005	.009		.004	.013	.018	.005	.015
e	.013	.007	**.877**		.016	.007	.008	.009	.014		.002	.005	.004	.003	.014
f	.009	.018	.019		.004	.007	.019	.016	.013		.017	.012	.007	.012	.000
g	.017	.017	.014		.014	.002	.009	.008	.009		.004	.014	.017	.014	.009
h	.019	**.793**	.019		.012	.002	.013	.002	.000		.011	.005	.007	.010	.001
i	.001	.007	.013		.008	.020	**.778**	.013	.005		.016	.007	.019	.018	.016
j	.007	.007	.020		.001	.004	.014	.010	.003		.012	.002	.005	.005	.004
k	.006	.016	.009		.013	.007	.010	.014	**.830**		.014	.003	.002	.013	.017
l	.007	.009	.004		.006	.007	.019	.013	.014		.019	.014	.018	.013	.004
m	.001	.005	.006		.007	.010	.010	.012	.007		.019	.005	.013	.009	.019
n	.007	.013	.001		.000	.010	.008	.009	.013		.014	.017	.013	.007	**.876**
o	.011	.012	.001		.010	.005	.005	.011	.010		.011	.018	**.779**	.000	.001
p	.010	.007	.000		.015	.000	.010	.014	.007		.002	.016	.007	.012	.004
q	.016	.005	.018		**.631**	.013	.018	.005	.005		.010	.006	.014	.014	.011
r	.005	.004	.002		.003	.005	.003	.013	.011		.004	**.530**	.008	.011	.013
s	.003	.018	.015		.018	.005	.007	.002	.002		.012	.019	.015	.009	.004
t	**.890**	.020	.019		.009	.014	.013	.000	.020		.018	.002	.003	.002	.005
u	.016	.014	.009		.004	**.678**	.005	.002	.007		.003	.017	.005	.009	.004
v	.007	.011	.016		.006	.006	.005	.010	.006		.004	.019	.000	.017	.019
w	.020	.015	.001		.003	.017	.007	.019	.011		.015	.003	.016	**.664**	.017
x	.009	.019	.014		.014	.008	.018	.001	.018		.013	.007	.004	.019	.017
y	.010	.003	.016		.010	.000	.015	.010	.015		.017	.013	.012	.003	.011
z	.003	.018	.014		.019	.016	.004	.018	.010		.018	.006	.007	.019	.004

character errors

	t	h	e		q	u	i	c	k		b	r	o	w	n
a	.017	.013	.007		.001	.007	.004	.009	.000		.014	.004	.001	.019	.009
b	.003	.001	.001		.005	.016	.012	.019	.003		**.670**	.014	.001	.020	.001
c	.000	.006	**.670**		.015	.013	.000	**.798**	.005		.007	.017	.018	.014	.018
d	.005	.017	.007		.006	.011	.011	.005	.007		.011	.000	.009	.012	.001
e	.015	.008	.230		.012	.014	.002	.013	.013		.003	.013	.019	.018	.007
f	.019	.004	.014		.016	.002	.008	.014	.007		.005	.001	.013	.016	.012
g	.019	.007	.012		.009	.019	.013	.004	.007		.006	.017	.008	.019	.010
h	.012	**.793**	.012		.009	.010	.019	.008	.014		.013	.005	.016	.000	.019
i	.003	.006	.020		.015	.002	**.778**	.012	.015		.009	**.550**	.004	.004	.007
j	.017	.007	.014		.002	.004	.002	.001	.015		.014	.013	.004	.004	.012
k	.013	.018	.005		.017	.014	.005	.016	**.830**		.016	.011	.015	.012	.013
l	.007	.007	.001		.011	.010	.009	.008	.017		.003	.010	.001	.005	.011
m	.016	.020	.005		.000	.004	.017	.013	.002		.001	.010	.016	.007	.006
n	.012	.017	.001		.004	.010	.008	.010	.011		.019	.000	.000	.007	**.876**
o	.018	.003	.017		**.540**	.019	.009	.009	.015		.008	.013	**.779**	.015	.012
p	.010	.008	.014		.011	.018	.010	.018	.004		.002	.015	.005	.006	.005
q	.013	.016	.017		.330	.013	.017	.017	.020		.014	.000	.017	.013	.009
r	.006	.008	.017		.019	.003	.009	.009	.011		.003	.440	.000	.009	.013
s	.018	.013	.015		.013	.018	.001	.019	.012		.017	.015	.004	.014	.013
t	**.890**	.004	.008		.013	.001	.001	.002	.007		.009	.018	.005	.013	.007
u	.014	.004	.011		.014	**.678**	.013	.004	.012		.006	.019	.005	.005	.020
v	.004	.017	.017		.019	.001	.011	.012	.005		.019	.005	.006	.020	.006
w	.013	.009	.005		.018	.001	.012	.012	.005		.003	.006	.018	**.664**	.007
x	.018	.005	.006		.003	.010	.005	.007	.006		.003	.006	.007	.013	.018
y	.012	.012	.003		.018	.011	.009	.011	.020		.014	.009	.001	.006	.001
z	.006	.009	.006		.005	.016	.004	.010	.001		.007	.019	.007	.013	.006

Bayes formula

$$P(c|x) = \frac{P(x|c)\,P(c)}{p(x)}$$

$$P(w|x) = \prod P(c_i|x)$$

$$P(w) \neq \prod P(c_i)$$

$$P(w|x) = \prod P(c_i|x)\,\frac{P(w)}{\prod P(c_i)}$$

(priors as used by classifier)

simple statistical language model

- take the per-character probabilities for each word
- adjust by word probabilities according to Bayes formula
- pick the word with the highest posterior probability
- spell correction: $P(w) = 1/N$ if word in dictionary, 0 otherwise

open issues

- **how do we deal with segmentation variants “clam” vs “dam”?**
- **how do we compute this efficiently?**
- **let's take a more general approach...**

statistical language models

statistical language models

- **statistical language models assign probabilities to string**
- **P(s) = ...**

unigram model

$$P(s) = \prod_i P(w_i)$$

- look up the probability (=normalized frequency) of each word in the string
- multiply together

bigram model

$$P(s)=\prod_i P(w_i|w_{i-1})$$

- look up the probability of each word in the string, conditional on the word that precedes it
- multiply together

n-gram model

$$P(s) = \prod_i P(w_i | w_{i-1} \ldots w_{i-n+1})$$

- **look up the probability of each word in the string, conditional on the n-1 words that precede it**
- **multiply together**

uni-/bi-/tri-gram models

- logical are as are confusion a may right tries agent goal the was diesel more object then information-gathering search is
- planning purely diagnostic expert systems are very similar computational approach would be represented compactly using tic-tac-toe a predicate
- planning and scheduling are integrated the success of naive bayes model is just a possible prior source by that time

probability estimates

- **uni/bi/trigram probabilities**
  - take large corpus of text
  - count # occurences of uni/bi/trigram
  - divide by total number of uni/bi/trigrams
- **there you have your probability...**

do you?

- “dwarf planet”
- “dark energy”
- “hockey mom”
- “drill baby drill”

sparsity

- **combinations**
  - 15000 common words
  - 225 million word pairs
  - 3.3 trillion word triples
- **naive assumption**
  - not found = probability 0 = can't occur
- **doesn't work... need non-zero probability for unseen n-grams**

add-one-smoothing

- **usual estimate:**
  - p = # occurrences / # total occurrences
- **add-one estimate**
  - p = (# occurences + 1) / (# total occurences + # classes)
- **properties**
  - uniform prior in a Bayesian sense
  - converges to true estimate in the case of large numbers
  - all probabilities non-zero
  - can use values other than 1 (e.g., 0.5)
  - doesn't work all that well for language modeling

linear interpolation smoothing

$$P(w_i|w_{i-1},w_{i-2})=c_3\hat{P}(w_i|w_{i-1},w_{i-2})+c_2\hat{P}(w_i|w_{i-1})+c_1\hat{P}(w_i)+c_0$$

- **final estimate is linear combination of uni/bi/trigram estimates**
- **pick the c_i empirically to maximize overall system performance (e.g., best OCR error rate)**

language model evaluation

- **good language model =**
  maximize the likelihood assigned to real texts
- **evaluate by computing P(s) on some test text s**
- **# bits / word = $-\log_2 P(\text{words}) / \#\text{words}$**
- **perplexity = $2^{\#\text{bits / word}}$**
  - average # of choices following a given word
  - “branching factor”

evaluation of n-gram models

- **P(s)**
  - evaluate directly: iterate through words and multiply
- **arg max P(s)**
  - for OCR, we have a set of recognition alternatives
  - use a dynamic programming algorithm to pick the optimal string

n-gram models

integrate with n-gram model using dynamic programming

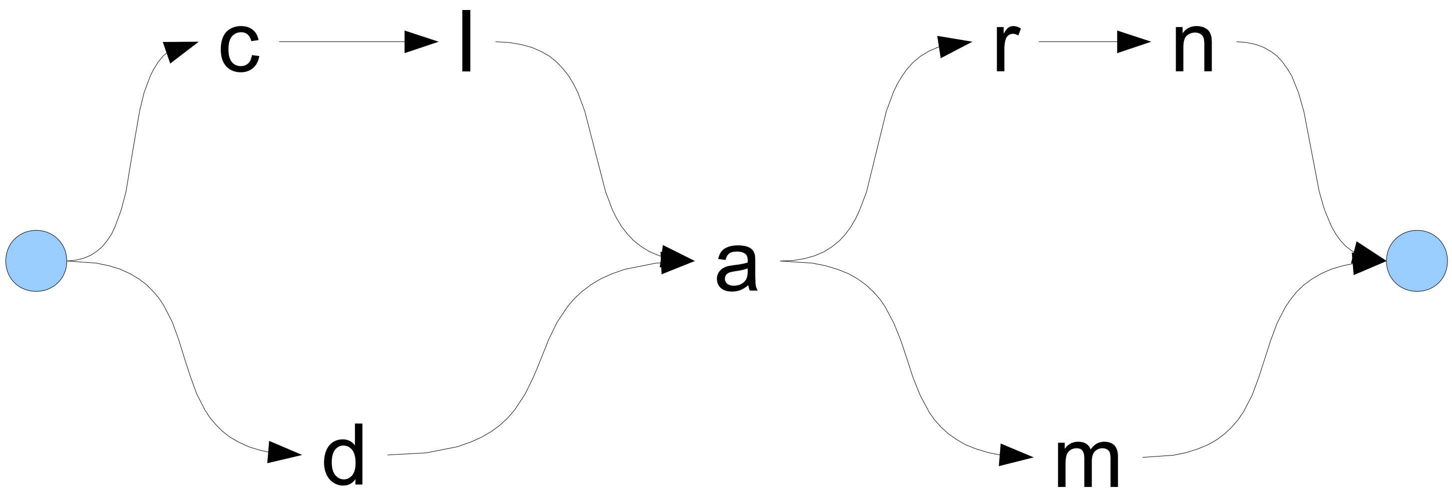
	t	h	e	q	u	i	c	k	b	r	o	w	n
a	.017	.013	.007	.001	.007	.004	.009	.000	.014	.004	.001	.019	.009
b	.003	.001	.001	.005	.016	.012	.019	.003	**.670**	.014	.001	.020	.001
c	.000	.006	**.670**	.015	.013	.000	**.798**	.005	.007	.017	.018	.014	.018
d	.005	.017	.007	.006	.011	.011	.005	.007	.011	.000	.009	.012	.001
e	.015	.008	.230	.012	.014	.002	.013	.013	.003	.013	.019	.018	.007
f	.019	.004	.014	.016	.002	.008	.014	.007	.005	.001	.013	.016	.012
g	.019	.007	.012	.009	.019	.013	.004	.007	.006	.017	.008	.019	.010
h	.012	**.793**	.012	.009	.010	.019	.008	.014	.013	.005	.016	.000	.019
i	.003	.006	.020	.015	.002	**.778**	.012	.015	.009	**.550**	.004	.004	.007
j	.017	.007	.014	.002	.004	.002	.001	.015	.014	.013	.004	.004	.012
k	.013	.018	.005	.017	.014	.005	.016	**.830**	.016	.011	.015	.012	.013
l	.007	.007	.001	.011	.010	.009	.008	.017	.003	.010	.001	.005	.011
m	.016	.020	.005	.000	.004	.017	.013	.002	.001	.010	.016	.007	.006
n	.012	.017	.001	.004	.010	.008	.010	.011	.019	.000	.000	.007	**.876**
o	.018	.003	.017	**.540**	.019	.009	.009	.015	.008	.013	**.779**	.015	.012
p	.010	.008	.014	.011	.018	.010	.018	.004	.002	.015	.005	.006	.005
q	.013	.016	.017	.330	.013	.017	.017	.020	.014	.000	.017	.013	.009
r	.006	.008	.017	.019	.003	.009	.009	.011	.003	.440	.000	.009	.013
s	.018	.013	.015	.013	.018	.001	.019	.012	.017	.015	.004	.014	.013
t	**.890**	.004	.008	.013	.001	.001	.002	.007	.009	.018	.005	.013	.007
u	.014	.004	.011	.014	**.678**	.013	.004	.012	.006	.019	.005	.005	.020
v	.004	.017	.017	.019	.001	.011	.012	.005	.019	.005	.006	.020	.006
w	.013	.009	.005	.018	.001	.012	.012	.005	.003	.006	.018	**.664**	.007
x	.018	.005	.006	.003	.010	.005	.007	.006	.003	.006	.007	.013	.018
y	.012	.012	.003	.018	.011	.009	.011	.020	.014	.009	.001	.006	.001
z	.006	.009	.006	.005	.016	.004	.010	.001	.007	.019	.007	.013	.006

n-gram

- **n-gram = sequence of n-things**
  - P(word | previous words)
  - P(character | previous characters)

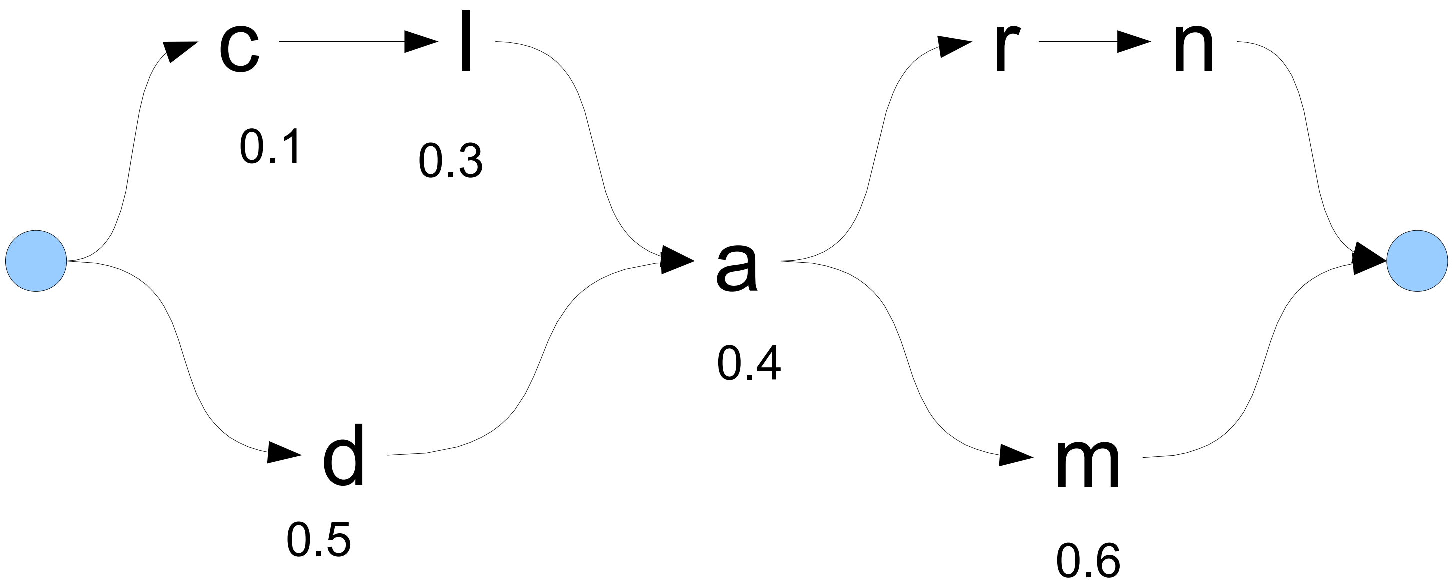
transducer-based language models

graphs as sets of strings



{“CLARN”, “DARN”, “CLAM”, “DAM”}

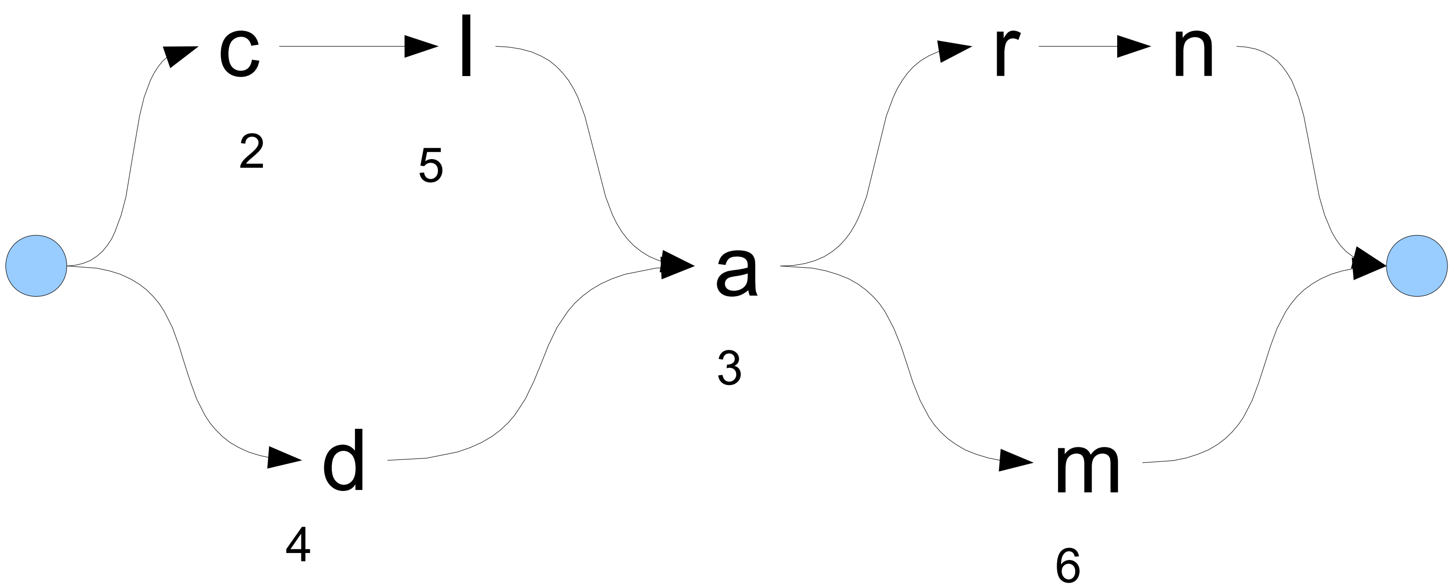
costs



P(clam) = 0.1 * 0.3 * 0.4 * 0.6

P(dam) = 0.5 * 0.4 * 0.6

costs



C(“clam”) = 2 + 5 + 3 + 6

C(“dam”) = 4 + 3 + 6

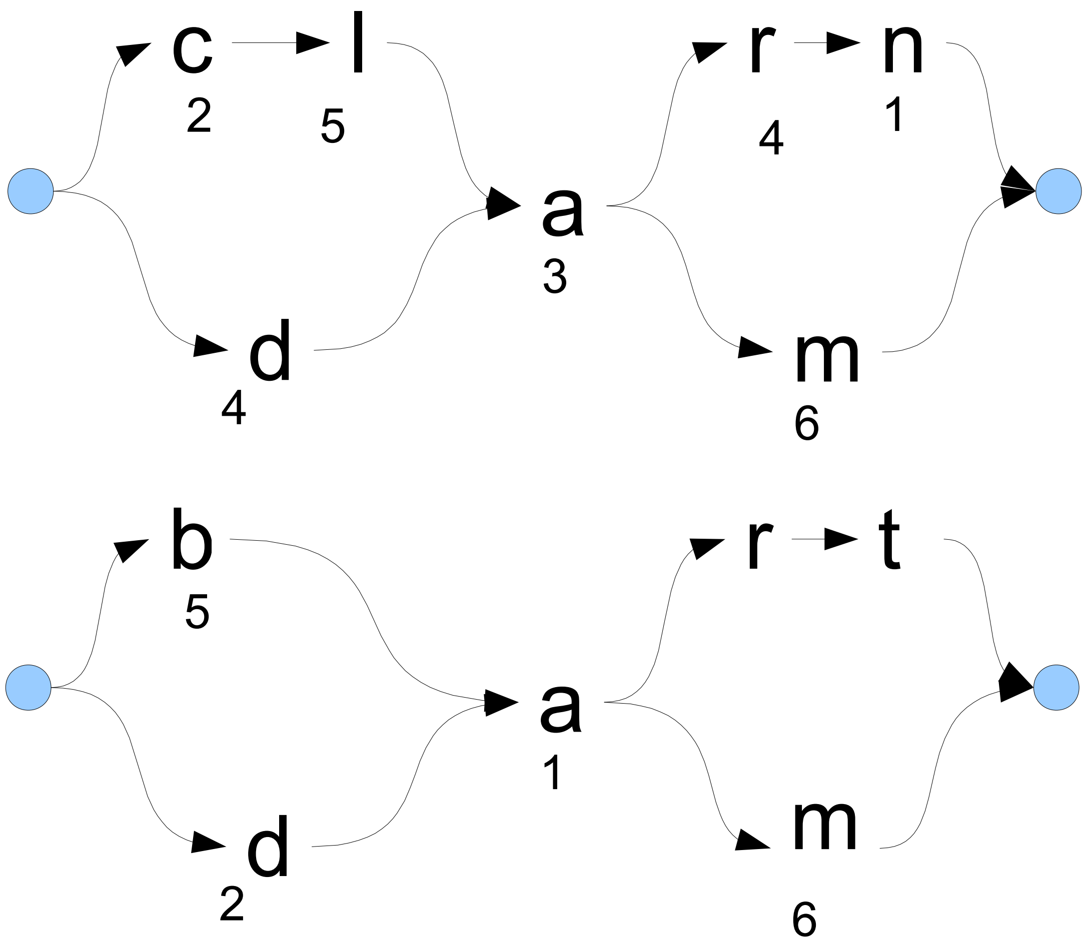
strings + weights

- labeled directed graphs = sets of strings
- cost of string = sum along path
- lowest cost path to ... = min over all paths
- (+,min) algebra
- equivalent to finite state acceptors / regular languages if costs are all 0 or infinity
- weighted finite state acceptors (special case of weighted finite state transducer)

language models

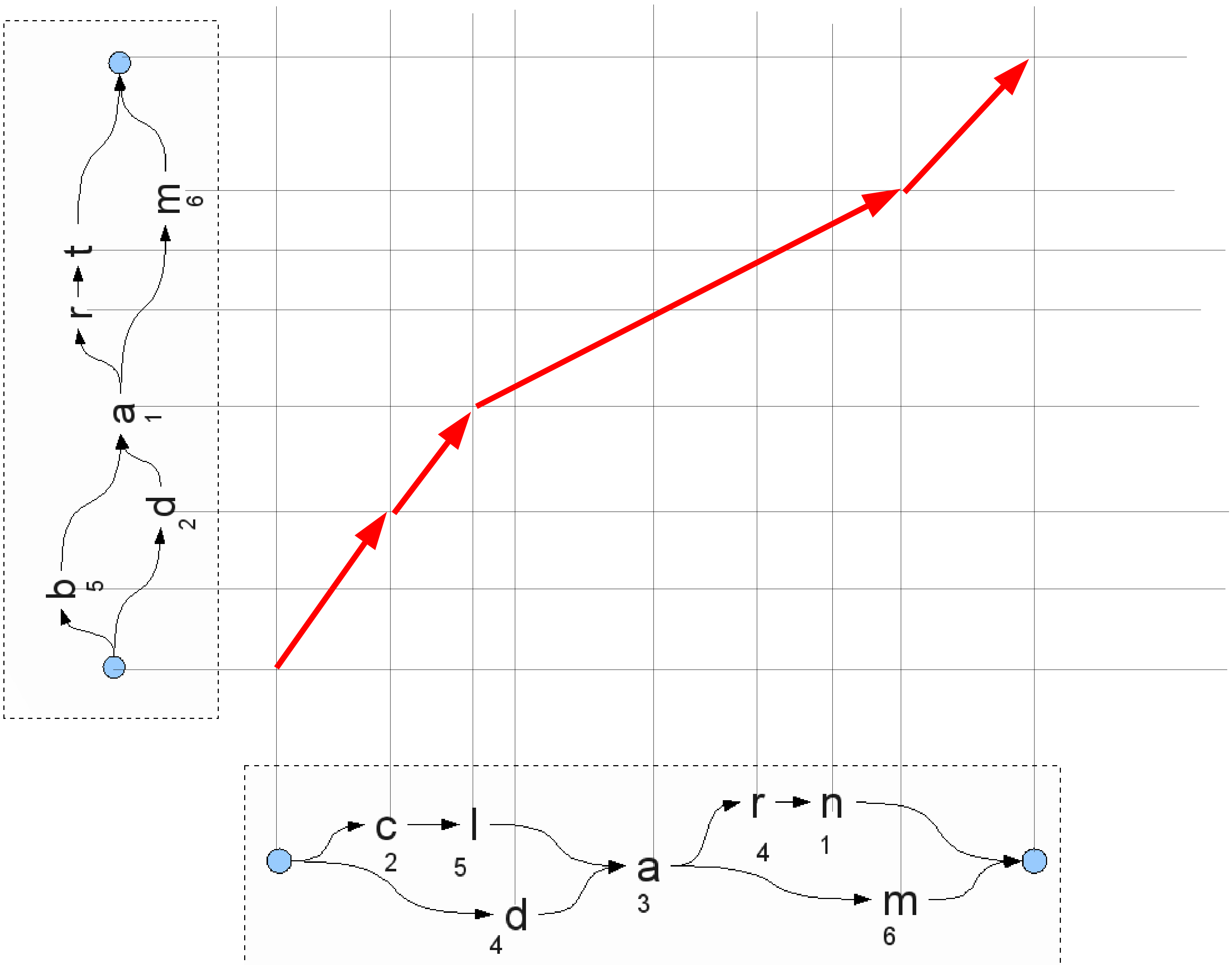
- **(properly normalized) weighted finite state transducers are statistical language models**
- **P(s) = exp(total cost along path)**
- **n-gram models (character or word) can be represented as weighted finite state transducers**

“intersection”



What string is possible within both graphs (transducers) and has the least total cost?

dynamic programming



OCR + language models

- **segmentation graph**
  - possible segmentations and classifications
  - posterior probabilities associated with each character
- **language model**
  - possible strings in the language
  - probabilities associated with each string
- **goal: find the best combination of the two**

solution

- **represent...**
  - segmentation graph as weighted finite state transducer
  - language model as finite state transducer
- **compute “intersection” (actually, “composition”)**

probabilistic formulation

Maximize over all strings:

$$P(\text{string}|\text{image}) = P(W|x)$$

Rewrite by summing over segmentations S:

$$P(W|x) = \sum_{S} P(W, S|x)$$

Bayes formula:

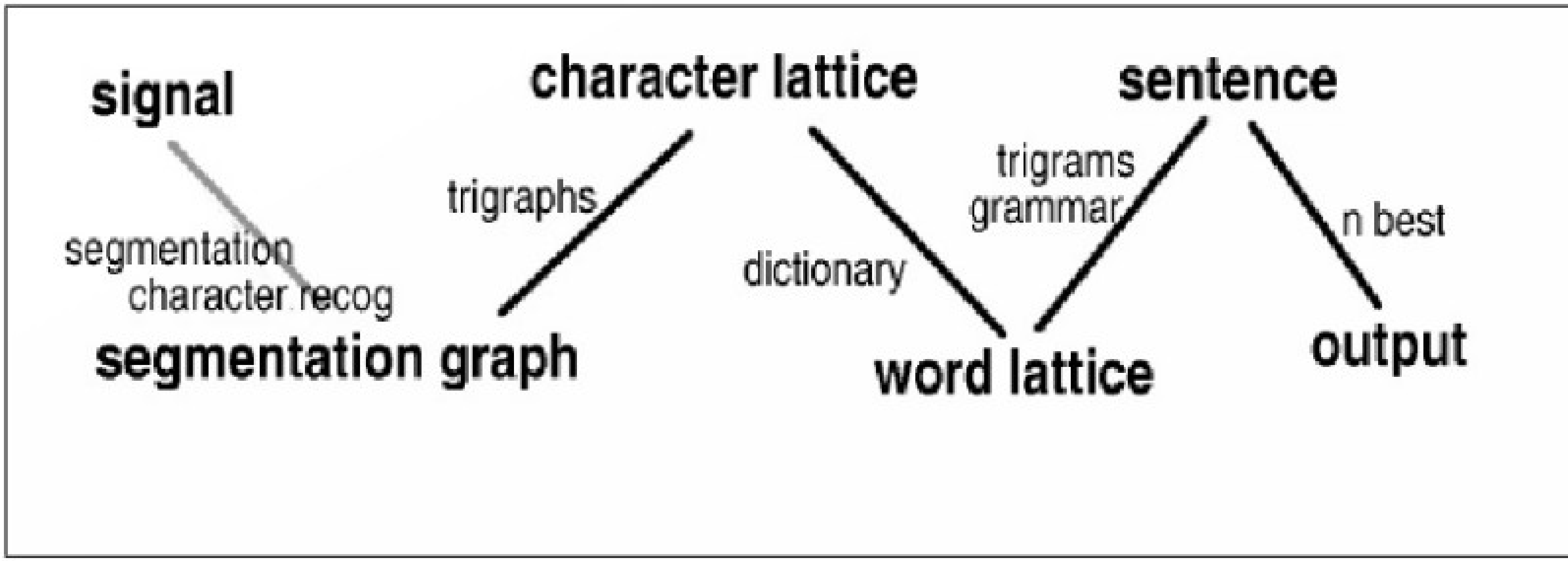
$$P(W, S|x) = \frac{P(x|W, S)\, P(W, S)}{P(x)}$$

Independence Assumption:

$$P(W, S|x) \approx P(W) \prod_{i} \frac{P(w_i|x_i)}{P(w_i)}\, P(S)$$

when $\mathrm{len}(W) = \mathrm{len}(S)$, 0 otherwise

algebraic manipulation of WFSTs



$$\text{recognizer} = \text{minimize}(\text{grammar} \circ \text{dictionary} \circ \text{trigraphs})$$

$$\text{solution} = \text{nbest}(\text{recognizer} \circ \text{segmentation graph})$$

summary

- OCR errors, ground truth
- OCR evaluation
- spell check, voting, confidence scores, ...
- statistical language models, n-grams
- smoothing
- perplexity
- weighted finite state transducers
- composition of WFSTs