Group Members: Cai Yang, Cui Jiayi, Nannan Wang,
(group 2) Yang Yang, Yu Wenting

Post-processing For OCR Data

Use topic models to improve OCR (D1,C5)

Content

PART I *Data Cleaning Process*

Load Data

Clean Text

Delete Uneven

- Comparing with ground-truth by rows
- Delete the uneven lines.
- Uneven: in our project means two lines don't equal number of characters.

PART II *Error Detection* *(garbage words)*

From Paper D1 Sectiom2.2

8 Error Detection Rules

01 A string composed of more than 20 symbols is garbage.

02 If the number of punctuation characters in a string is greater than the number of alphanumeric characters.

03 Ignoring the first and last characters in a string, if there are two or more different punctuation characters in the string

04 If there are three or more identical characters in a row in a string

05 If the number of uppercase characters in a string is greater than the number of lowercase characters, and if the number of uppercase characters is less than the total number of characters in the string

06 If all the characters in a string are alphabetic, and if the number of consonants in the string is greater than 8 times the number of vowels in the string, or vice-versa,

07 If there are four or more consecutive vowels in the string or five or more consecutive consonants in the string, it is garbage

08 If the first and last characters in a string are both lowercase and any other character is uppercase

PART III *Error Correction-Topic Models*

From Paper C5 Section 3.2

Step1 LDA Topic Models

Generate 30 topics for each documents and compute the probability

Step2 Calculate Confusion Matrix

Doing Fuzzy Matching

Generate Confusion Matrix

Step3 Compute Scores

Generate all possible word candidates

Compute scores for each candidates using step1 & step2

Step4 Correct

For each words select the highest score

Correct the words

PART III *Error Correction-Topic Models*

From Paper C5 Section 3.2

$$\mathrm{Score}(w_c) = P(w_c) \prod_j^N P(l_j^f | l_j^s)$$

$$P(w) = \sum_k^M P(w|t_k) P(t_k)$$

Probability of a word under a topic

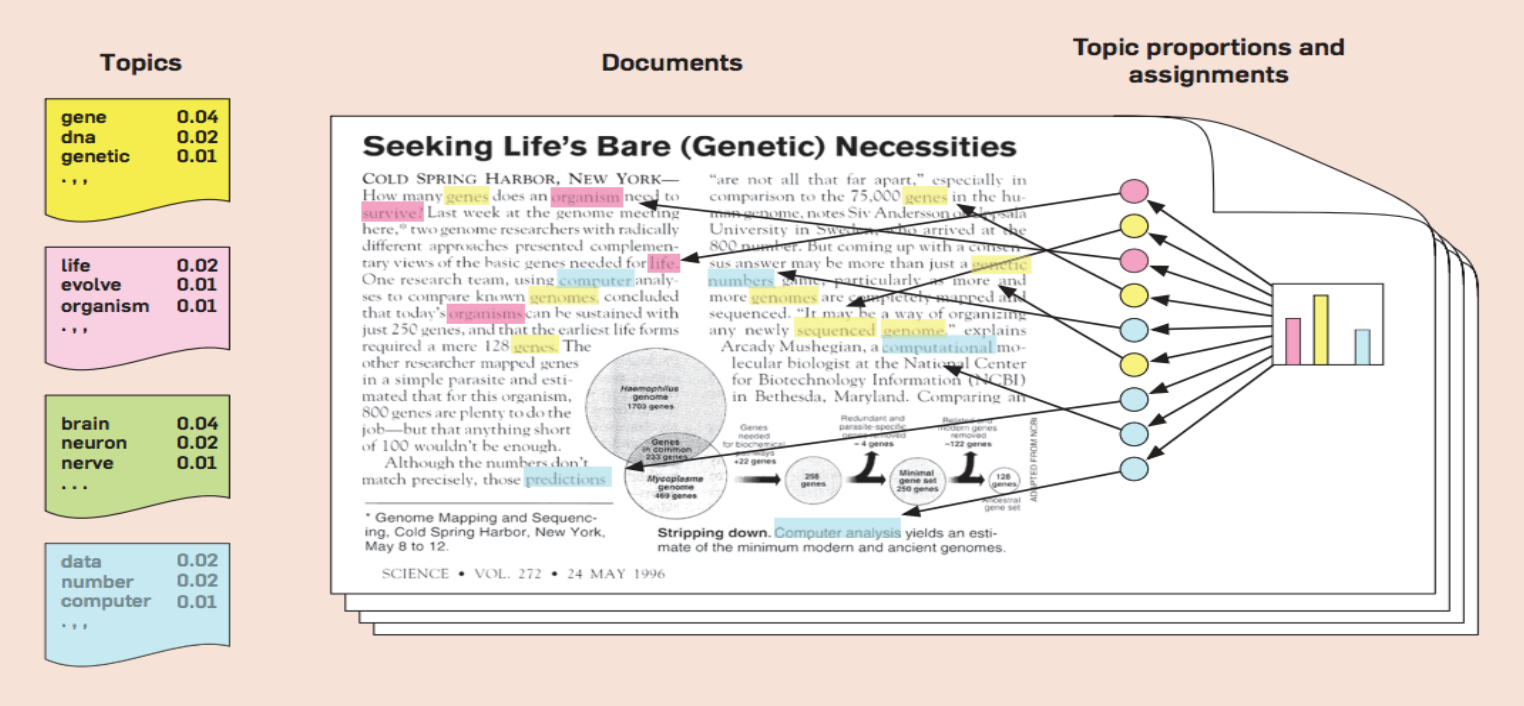
Probability of a topic in a document

$P(l_j^f | l_j^s)$ is the probability that letter l_j^s was mistaken for l_j^f

Step1 LDA Topic Models

$$P(w) = \sum_{k}^{M} P(w|t_k)P(t_k)$$

Figure 1. The intuitions behind latent Dirichlet allocation. We assume that some number of "topics," which are distributions over words, exist for the whole collection (far left). Each document is assumed to be generated as follows. First choose a distribution over the topics (the histogram at right); then, for each word, choose a topic assignment (the colored coins) and choose the word from the corresponding topic. The topics and topic assignments in this figure are illustrative—they are not fit from real data. See Figure 2 for topics fit from data.



LDA Topic Models

Train set: 80 documents

Test set: 20 documents

$$P(w) = \sum_{k}^{M} P(w|t_k)P(t_k)$$

P(W) -- Result

EXHIBIT	1.410324e-04	8.846296e-05	7.364094e-05	5.873865e-05	1.108232e-04
D	1.231148e-04	1.154478e-04	1.046156e-04	1.662151e-04	6.583606e-05
STAFF	4.647684e-05	4.590222e-05	2.177579e-05	4.548453e-05	5.449980e-05
REPORT	4.253964e-04	6.966216e-04	3.498882e-04	3.949201e-04	3.565690e-04
Ma	1.576632e-05	6.805707e-06	2.936210e-06	5.375728e-06	4.781064e-06
rch	1.576632e-05	6.805707e-06	2.936210e-06	5.375728e-06	4.781064e-06
1972	5.053519e-05	3.319093e-05	5.028957e-05	2.846623e-05	5.555214e-05
EXHIBIT	9.563508e-05	9.106156e-05	1.034338e-04	4.510108e-05	1.403734e-04
D	1.117193e-04	1.142945e-04	1.041921e-04	6.427003e-05	1.292587e-04
STAFF	6.617885e-05	5.329141e-05	2.870197e-05	4.474444e-05	3.796790e-05
REPORT	3.535541e-04	3.484049e-04	4.056158e-04	2.550544e-04	3.958782e-04
Ma	9.883238e-06	1.010469e-05	2.172914e-06	2.900113e-06	2.859871e-06
rch	9.883238e-06	1.010469e-05	2.172914e-06	2.900113e-06	2.859871e-06
1972	1.077024e-04	3.817442e-05	4.753034e-05	3.762594e-05	5.468387e-05
EXHIBIT	8.324368e-05	1.040504e-04	7.825063e-05	8.481069e-05	6.941038e-05
D	1.038833e-04	9.960531e-05	8.868838e-05	9.997992e-05	8.889652e-05
STAFF	1.480647e-05	4.940796e-05	8.757694e-06	4.271602e-05	8.525086e-06
REPORT	3.584698e-04	6.567046e-04	3.688413e-04	3.545804e-04	3.311007e-04
Ma	3.870949e-09	3.627672e-06	7.296715e-07	6.081534e-06	2.652052e-06
rch	3.870949e-09	3.627672e-06	7.296715e-07	6.081534e-06	2.652052e-06
1972	4.584370e-05	3.190681e-05	3.499714e-05	4.573325e-05	4.219801e-05
EXHIBIT	4.076456e-05	5.857666e-05	5.383475e-05	8.581376e-05	7.694486e-05
D	6.664174e-05	7.668749e-05	8.565761e-05	1.105256e-04	9.644555e-05
STAFF	1.548654e-05	5.620396e-05	8.138561e-06	2.534409e-05	3.718760e-05
REPORT	4.494475e-04	3.932211e-04	2.982990e-04	3.954002e-04	3.809855e-04
Ma	4.837995e-06	1.118029e-05	3.435429e-12	5.388202e-06	1.296698e-05
rch	4.837995e-06	1.118029e-05	3.435429e-12	5.388202e-06	1.296698e-05
1972	1.548822e-05	3.848495e-05	1.697548e-05	3.481470e-05	3.156877e-05

Step2 Confusion Matrix -- Fuzzy Matching

Step1

LD_similarity=1-(levenshtein distance)/(nchar of actual world)

##Levenshtein distance : the minimum number of deletions, insertions, or substitutions required to transform string_s into string_t

Step2

Choose the one with largest ***LD_similarity*** in ***CANDIDATE_NUM***(here we choose 3) of OCR words.

Step3

To match the actual word.

If none of these candidate has ***LD_similarity*** above ***SIM_THRES***(we choose 0.5), then give up this actual word.

We also give up the words which do not have the equal length

Step2 Confusion Matrix -- Fuzzy Matching

Ground Truth	OCR
Air	Alr
Quality	O
Committee	allity
	Commlttee

Alr : *LD = 1*
O : *LD = 4*
allity : *LD = 5*
Alr : *LD_similarity = 1-1/3 = 2/3 > 0.5*
O : *LD_similarity = 1- 4/3 = -1/3*
allity : *LD_similarity = 1-5/3 = 2/3*

Ground Truth	OCR
Air	Alr
Quality	O
Committee	allity
	Commlttee

O : *LD = 7*
allity : *LD = 3*
Commlttee : *LD = 8*
O : *LD_similarity = 1- 7/7= 0*
allity : *LD_similarity = 1- 3/7 = 4/7 > 0.5*
Commlttee : *LD_similarity = 1- 8/7= -1/7*

*Step2 Confusion Matrix -- results: 110 * 110 matrix*

110 strings :

```
  [1] "-"  "'"  "–"  "—"  "\f" "!"  "\"" "#"  "$"  "%"  "&"  "("  ")"  "*"  ","
 [16] "."  "/"  ":"  ";"  "?"  "["  "\\" "]"  "^"  "_"  "{"  "|"  "}"  "~"  "‘"
 [31] "’"  "“"  "”"  "¢"  "£"  "+"  "<"  "ﬁ"  "ﬂ"  "="  ">"  "«"  "»"  "§"  "®"
 [46] "°"  "•"  "0"  "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "a"  "A"  "b"
 [61] "B"  "c"  "C"  "d"  "D"  "e"  "E"  "é"  "f"  "F"  "g"  "G"  "h"  "H"  "i"
 [76] "I"  "j"  "J"  "k"  "K"  "l"  "L"  "m"  "M"  "n"  "N"  "o"  "O"  "p"  "P"
 [91] "q"  "Q"  "r"  "R"  "s"  "S"  "t"  "T"  "u"  "U"  "v"  "V"  "w"  "W"  "x"
[106] "X"  "y"  "Y"  "z"  "Z"
```

*Step2 Confusion Matrix -- results: 110 * 110 matrix*

Part of Matrix :

$$P(l_j^f \mid l_j^s)$$

0.9941036	1.89E-05	7.57E-05	0	0.001022166	0.000208219	0	0	0.001808	9.46E-06
9.93E-05	0.999007	0	0	0	9.93E-05	0	0	0	0
0	0	0.998337541	0	0	0	0	0	6.16E-05	0
0.0079727	0	0	0.9453303	0	0	0	0.001139	0	0.001708428
0	0	2.16E-05	2.16E-05	0.995407008	0.003212938	6.47E-05	2.16E-05	4.31E-05	0
0	9.93E-05	0	0	0.027209533	0.972095333	0	0	0	0
0	0	0	0	4.03E-05	0	0.999737924	0	0	0
0	0	0	0	0	0	0	0.7914206	0	0
1.21E-05	0	0	0	0.000363192	1.21E-05	6.05E-06	0	0.999425	0
0.0020937	0	0	0.0093413	0	0.00048317	0	0.0011274	0.046706	0.922853922
0	0	0	0	0	0	0	0	0.333333	0
0	0	3.69E-05	0	0	0	7.38E-05	0	3.69E-05	0
0	0.000543	0	0	0	0.000543183	0	0	0	0
0	0	0	0	8.41E-05	0	0	0	0	0
0	0	0	0	0.002925688	0.007606788	0	0	0	0
0	0	0.000725581	0	0	0	0	0	2.13E-05	0
0	0	0.000837054	0.000558	0	0	0	0.000279	0	0.000279018
0	0	0	0	0	0	0.000314994	0	0	0
0	0.000193	0.000192567	0	0	0.001733102	0	0.0025034	0	0
0	0	0	0	0	0	0	0.0007782	0	0
0	0	0	0	0.000952381	0	0	0	0	0
0	0	0	0	0	0	0	0	0.00041	0
0	0	0	0	0	0	0	0	0	0
1.61E-05	0	8.04E-06	0	0	7.23E-05	0	3.22E-05	3.22E-05	1.61E-05
0	0.000563	0.00056338	0	0	0	0	0	0	0
8.49E-05	0	0	0	2.83E-05	0	2.83E-05	2.83E-05	5.66E-05	2.83E-05
0	0	0	0	0	0.000407166	0	0	0	0
0	0	0.000266388	0	6.91E-05	0	3.95E-05	0.0001085	2.96E-05	0
0	0	0	0	0	0	0	0	0	0
5.81E-05	0	0	9.69E-06	0	0	0	0	1.94E-05	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0.000244439	0	0	0	0.000244439
0	0	0	0	0	0	0	0	0.000685	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	2.32E-05	0	4.64E-05	0	4.64E-05	0
0	0	0	0	0	0	0	0	0	0.010419251
0.0006944	0	0	4.87E-05	0.000682186	0	1.22E-05	0	0.001206	0
0	0	0	0	0	0	0	0	0	0
3.36E-05	0	0	0	0.000151352	0	0.000496098	0	8.41E-06	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	2.66E-05	0.0009322	0	0
0	0	0	0	0	0	0	0.0023041	0	0

Step3 Compute all candidates Scores

(generate all possible candidates) $P(w_c)$ + bias(1+e5)

Step4 Select the best candidates *(the highest score)*

Truth	Tesseract	Processed Tesseract
exhibit	exhlblt	exhibit
driver	driver	driver
before	before	before
in	1n	an
last	last	last

PART IV *Performance Evaluation*

Step1

Find unique words in two texts separately. And only the ones that appear in both texts are used.

Use unique words as anchors to split each segment into smaller ones

Step2

Delete unique words from all segments (delete the first and the last word in every segments)

Step3

Use levenshtein distance to calculate the number of incorrect characters in every segments

The distance gives the minimal possibly weighted number of insertions, deletions and substitutions needed to transform one string into another.

Step4

Calculate precision and recall

Recall: number of correct items/number of items in ground truth

PART IV *Performance Evaluation*

	Tesseract	Tesseract with (detection)	Tesseract with (correction)
Word_wise_recall	0.6245921	0.5822522	0.5868309
Word_wise_precision	0.6160689	0.7250324	0.7087555
character_wise _recall	0.72467	0.48775	0.57566
character_wise _precision	0.71864	0.47854	0.56715

Thanks for Listening !

Have a nice day!