Project 4 Predictive Modeling

Optical Character Recognition

Group 2

Index

- Project Goal
- Error Detection
- Feature Extraction
- Model: Adaboost
- Evaluation
- More Thoughts…

PROJECT GOAL

Pre-processing and Word recognition:

1. converting scanned images into machine readable character streams
2. process raw scanned images relying on the Tessearct OCR machine

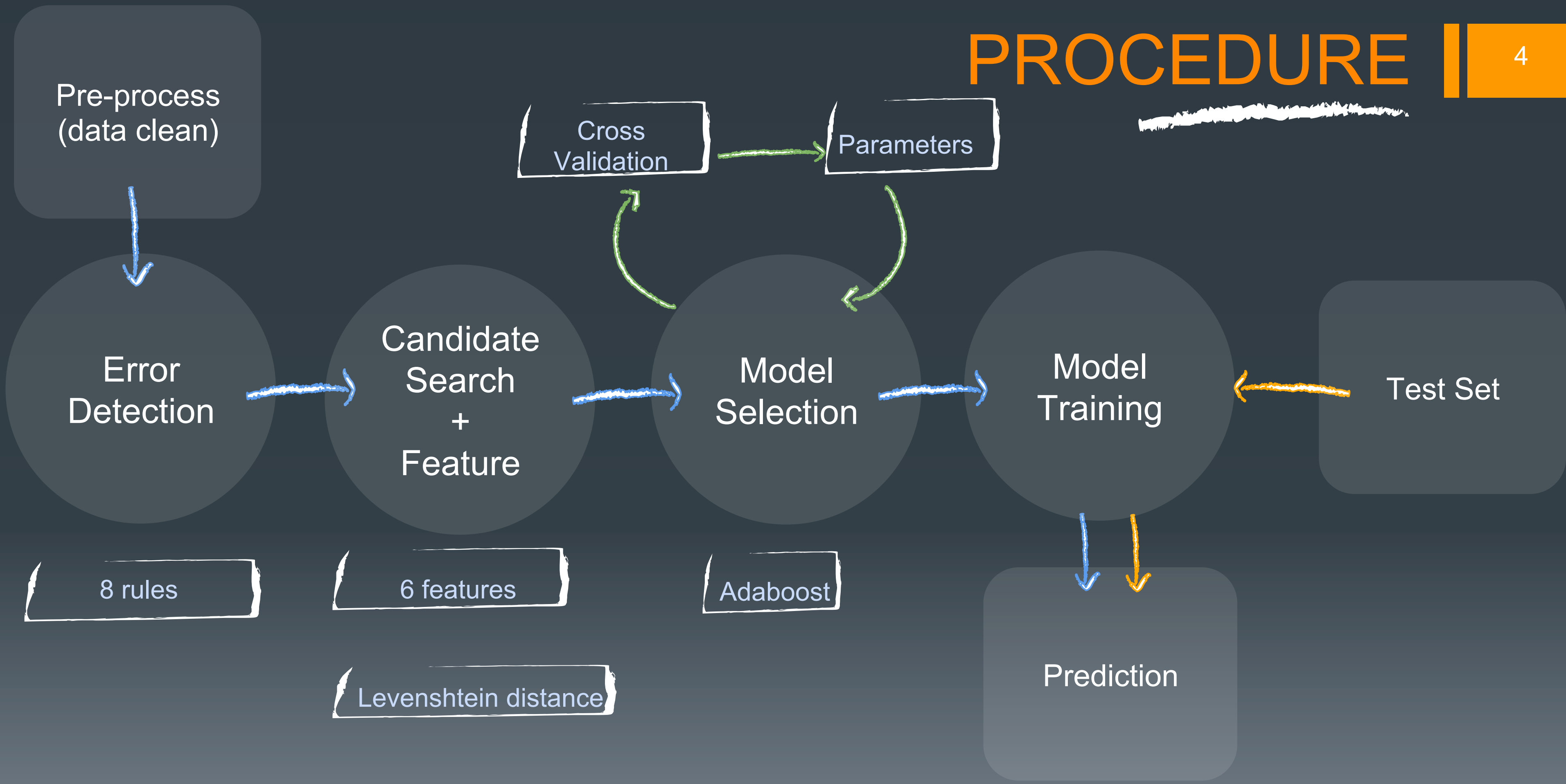
Methods:

1. Rule-based techniques
2. Supervised model – correction regressor

Evaluation on:

1. Precision : is the percentage of correctly found words with respect to the total word count of the OCR output
2. Recall : the percentage of words in the original text correctly found by the OCR engine

PROCEDURE



Pre-process (data clean)
Error Detection
8 rules
Candidate Search + Feature
6 features
Levenshtein distance
Cross Validation
Parameters
Model Selection
Adaboost
Model Training
Test Set
Prediction

Error Detection

- Example: iiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiii...
- Example: ?3//la‘.
- *Example: b?bl@bjk.1e.322*
- *Example: aaaaaBlE*
- *Example: BBEYaYYq*
- *Example: jabwqbpP*
- *Example: buauub*
- *Example: awwgrapHic*

	Tesseract	Ground_Truth
0	1nd1v1duals	individuals
1	representlng	representing
2	you	your
3	companles	companies
4	1ndustry	industry
5	comm1ttee	committee
6	companles	companies
7	placlng	placing
8	1ncrease	increased
9	communlcatlons	communications
10	skllls	skills
11	senlor	senior
12	on	On
13	medla	media
14	reltlons	relations
15	1n	in

FEATURE

- Levenshtein edit distance

$$score(\mathrm{w}_c, \mathrm{w}_e) = 1 - \frac{dist(\mathrm{w}_c, \mathrm{w}_e)}{\delta + 1}$$

- String similarity

$$\begin{aligned} & score(\mathrm{w}_c, \mathrm{w}_e) \\ & \quad = \alpha_1 \cdot nlcs(\mathrm{w}_c, \mathrm{w}_e) + \alpha_2 \cdot nmnlcs_1(\mathrm{w}_c, \mathrm{w}_e) \\ & \quad + \alpha_3 \cdot nmnlcs_n(\mathrm{w}_c, \mathrm{w}_e) + \alpha_4 \cdot nmnlcs_z(\mathrm{w}_c, \mathrm{w}_e). \end{aligned}$$

- *Language popularity*

$$score(\mathrm{w}_c, \mathrm{w}_e) = \frac{freq_1(\mathrm{w}_c)}{\max_{\mathrm{w}'_c \in C} freq_1(\mathrm{w}'_c)}.$$

- *Lexicon existance*

$$score(\mathrm{w}_c, \mathrm{w}_e) = \begin{cases} 1 & \text{if } \mathrm{w}_c \text{ exists in the lexicon} \\ 0 & \text{otherwise} \end{cases}$$

- *Exact-context popularity*

$$score(\mathrm{w}_c, \mathrm{w}_e) = \frac{\sum_{\mathbf{c} \in \mathcal{G}_c} freq_n(\mathbf{c})}{\max_{\mathrm{w}'_c \in C} \{\sum_{\mathbf{c}' \in \mathcal{G}'_c} freq_n(\mathbf{c}')\}}$$

- *Relaxed-context popularity*

Candidate Search

$$\{ w_c \,|w_c \in \mathcal{L}, dist(w_c, w_e) \leq \delta\},$$

- Candidate Search Results:

	Typo	Truth	Candidate	led_score	ss_score	lp_score	Label	predicted_confidence
0	willlam	william	will	0.25	1.636364	1.000000	0	0.418665
1	willlam	william	willful	0.25	1.607143	0.001712	0	0.082236
2	willlam	william	william	0.75	2.142857	0.039954	1	0.636513
3	willlam	william	williams	0.50	1.866667	0.003995	0	0.095767
4	willlam	william	willing	0.25	1.285714	0.006849	0	0.056328
5	willlam	william	wills	0.25	1.500000	0.000571	0	0.056328
6	nvolvng	involving	cooling	0.25	0.857143	0.011494	0	0.026191
7	nvolvng	involving	evolve	0.25	0.961538	0.045977	0	0.027614
8	nvolvng	involving	evolved	0.25	0.892857	0.034483	0	0.027614
9	nvolvng	involving	evolves	0.25	0.892857	0.011494	0	0.027614
10	nvolvng	involving	evolving	0.50	1.633333	0.057471	0	0.108891
11	nvolvng	involving	involvad	0.25	0.833333	0.011494	0	0.026191
12	nvolvng	involving	involve	0.25	0.892857	0.126437	0	0.027614
13	nvolvng	involving	involved	0.25	0.833333	1.000000	0	0.026191
14	nvolvng	involving	involves	0.25	0.833333	0.160920	0	0.026191
15	nvolvng	involving	involve	0.25	0.833333	0.011494	0	0.026191
16	nvolvng	involving	involving	0.50	1.656250	0.505747	1	0.609977

Adaboost

- AdaBoost (with decision trees as the weak learners) is often referred to as the best out-of-the-box classifier.
- When used with decision tree learning, information gathered at each stage of the AdaBoost algorithm about the relative 'hardness' of each training sample is fed into the tree growing algorithm such that later trees tend to focus on harder-to-classify
- AdaBoost is sensitive to noisy data and outliers. In some problems it can be less susceptible to the overfitting problem than other learning algorithms.

Adaboost

Parameter Tuning


```
In [70]: for mean_score, params in zip(cvres["mean_test_score"], cvres["params"]):
             print(mean_score, params)

0.40374915824108776 {'learning_rate': 0.01, 'loss': 'linear', 'n_estimators': 50}
0.3680232694914383 {'learning_rate': 0.01, 'loss': 'linear', 'n_estimators': 100}
0.4023762621878514 {'learning_rate': 0.01, 'loss': 'square', 'n_estimators': 50}
0.34098242730811906 {'learning_rate': 0.01, 'loss': 'square', 'n_estimators': 100}
0.4083847633391772 {'learning_rate': 0.01, 'loss': 'exponential', 'n_estimators': 50}
0.39643609293192866 {'learning_rate': 0.01, 'loss': 'exponential', 'n_estimators': 100}
0.15414374962757887 {'learning_rate': 0.05, 'loss': 'linear', 'n_estimators': 50}
-0.23349487386235154 {'learning_rate': 0.05, 'loss': 'linear', 'n_estimators': 100}
-0.10284196383632471 {'learning_rate': 0.05, 'loss': 'square', 'n_estimators': 50}
-0.9978515475452031 {'learning_rate': 0.05, 'loss': 'square', 'n_estimators': 100}
0.27868712935106005 {'learning_rate': 0.05, 'loss': 'exponential', 'n_estimators': 50}
-0.05210748438201115 {'learning_rate': 0.05, 'loss': 'exponential', 'n_estimators': 100}
-0.18220429066384375 {'learning_rate': 0.1, 'loss': 'linear', 'n_estimators': 50}
-0.3433878616741871 {'learning_rate': 0.1, 'loss': 'linear', 'n_estimators': 100}
-0.9308229281848934 {'learning_rate': 0.1, 'loss': 'square', 'n_estimators': 50}
-1.0131045582770672 {'learning_rate': 0.1, 'loss': 'square', 'n_estimators': 100}
-0.06313284282744469 {'learning_rate': 0.1, 'loss': 'exponential', 'n_estimators': 50}
-0.7592209661142575 {'learning_rate': 0.1, 'loss': 'exponential', 'n_estimators': 100}
-0.31279929981489774 {'learning_rate': 0.3, 'loss': 'linear', 'n_estimators': 50}
-0.37670143401303086 {'learning_rate': 0.3, 'loss': 'linear', 'n_estimators': 100}
-1.0867650054545688 {'learning_rate': 0.3, 'loss': 'square', 'n_estimators': 50}
-0.976816075191342 {'learning_rate': 0.3, 'loss': 'square', 'n_estimators': 100}
-1.0901767341119117 {'learning_rate': 0.3, 'loss': 'exponential', 'n_estimators': 50}
-1.5741717968173896 {'learning_rate': 0.3, 'loss': 'exponential', 'n_estimators': 100}
0.37608463662658714 {'learning_rate': 1, 'loss': 'linear', 'n_estimators': 50}
0.38336523445017 {'learning_rate': 1, 'loss': 'linear', 'n_estimators': 100}
0.38065389767817553 {'learning_rate': 1, 'loss': 'square', 'n_estimators': 50}
```

Adaboost

Parameter Tuning


```
ada_grid_search_fit.best_estimator_
```

```
AdaBoostRegressor(base_estimator=None, learning_rate=0.01, loss='exponential',
         n_estimators=50, random_state=None)
```

EVALUATION

- Recall is the percentage of words in the original text correctly found by the OCR engine
- Precision is the percentage of correctly found words with respect to the total word count of the OCR output

	Measure	Tesseract	Post
0	word_wise_recall	0.645514	0.737842
1	word_wise_precision	0.651473	0.744653
2	character_wise_recall	0.921204	0.941582
3	character_wise_precision	0.950425	0.969046

If we have more time …

- Larger vocabulary dictionary
- Contextual constraints
- *Study other methods, e.g ngrams, nn*

Thanks!

Refence

1. (C2) Mei, Jie, et al. Statistical Learning for OCR Text Correction. School of Computing and Informatics, University of Louisiana at Lafayette, 2016.

2. (D1) Kulp, Scott, and April Kontostathis. On Retrieving Legal Files: Shortening Documents and Weeding Out Garbage. Collegeville, Department of Mathematics and Computer Science, Ursinus College.