Code Documentation: Automatic OCR Error Correction Using Modified Levenshtein Distance Metric and Language Model

Ruichi Yu

ry2254@columbia.edu

System Design

The system for Automatic OCR Error Correction Using Modified Levenshtein Distance Metric and Language Model has three major parts:

1. Dictionary construction.
2. Error pattern mining.
3. Error detection and correction.

Environment

Codes are written in python and C#.

For python part, all codes are running on Linux server that has nltk 2.0, inflect, Mysql-python installed.

Scheduled Tasks

All of the codes are running as scheduled tasks on server.

To check the scheduled tasks, use command
sudo crontab –l

To modify the scheduled tasks, use command
sudo crontab –e

All codes are running one time an hour. However, since some of the codes may need more than one hour to finish, we have a configuration table on our databases to store the configurations and lock information of each program.

TranscriptID: The ID of video caption that already has a transcript dictionary.
DailyTime: The date that already has a time based dictionary.
EventID: The ID of event that already has an event dictionary.
Lock_Transcript: 1 if transcript dictionary construction code is running, 0 otherwise.
Lock_Event: 1 if event dictionary construction code is running, 0 otherwise.
Lock_Daily: 1 if time based dictionary construction code is running, 0 otherwise.
Lock_MineError: 1 if code of mining error is running, 0 otherwise.
Lock_GoogleSearch: 1 if code of Google search is running, 0 otherwise.

Lock_Correction: 1 if code of error detection and correction is running, 0 otherwise.
CaptionID： The ID of video caption that already has an Time based dictionary for the corresponding date.

Dictionary Construction

Three dictionaries are constructed with codes in this section:
1. Video Transcripts Dictionary (VT-Dict)
2. Linked Event Dictionary (LE-Dict)
3. Time-Based Dictionary (TB-Dict)

Location of Code

Server: orbit.cs.columbia.edu

Location of code:
/home/ruichi/Documents/OCR/Transcript_Release (VT-Dict)
/home/ruichi/Documents/OCR/Event_Release (LE-Dict)
/home/ruichi/Documents/OCR/Release (TB-Dict)

Location of dictionary:
/home/ruichi/Documents/OCR/Dictionary/Transcripts (VT-Dict)
/home/ruichi/Documents/OCR/Dictionary/EventDict (LE-Dict)
/home/ruichi/Documents/OCR/Dictionary/DailyDict (TB-Dict)

Location of configuration:
/home/ruichi/Documents/OCR/App.config (VT-Dict)
/home/ruichi/Documents/OCR/App.config (LE-Dict)
/home/ruichi/Documents/OCR/App.config (TB-Dict)

Video Transcripts Dictionary

Keys in this code:

"Configuration": The database connection of configuration table.
"OCR_Caption": The database connection of video caption table.
"Transcripts”: The database connection of video transcripts.

“GNewsCrawler”: The database connection of linked event.

The main function of this code is to create transcript dictionary for each video caption.

Linked Event Dictionary

Keys in this code:

"Configuration": The database connection of configuration table.
"OCR_Caption": The database connection of video caption table.
"Transcripts”: The database connection of video transcripts.
“GNewsCrawler”: The database connection of linked event.
“Max_articles”: The maximum number of articles we used to create an event dictionary of a certain event.

The main function of this code is to create event dictionary for each linked event.

Time-Based Dictionary

Keys in this code:

"Configuration": The database connection of configuration table.
"OCR_Caption": The database connection of video caption table.
"Transcripts”: The database connection of video transcripts.
“GNewsCrawler”: The database connection of linked event.
“Max_articles”: The maximum number of articles we used to create a time based dictionary.
“Max_trans”: The maximum number of transcripts we used to create a time based dictionary.

The main function of this code is to create time based dictionary for each time slot.

Error Pattern Mining

There are two codes in this section:

MineError_Realtime.py:
Mine frequent errors from all video captions.

Google_Search_Realtime.py:
Utilize Google search to mine special words.

Location of Code

Server: orbit.cs.columbia.edu

Location of code:
/home/ruichi/Documents/OCR/ MineError_Realtime.py
/home/ruichi/Documents/OCR/ Google_Search_Realtime.py

The databases connection configurations are in:
/home/ruichi/Documents/OCR/Dictionary/DB_Config.txt

MineError_Realtime.py

This code is used to mine frequent errors from all captions.

The mined errors are stored in:
'/home/ruichi/Documents/OCR/Dictionary/Mine.txt'

Google_Search_Realtime.py

This code is used to utilize Google search to mine special words.

The mined special words are stored in:
/home/ruichi/Documents/OCR/Dictionary/AfterMine.txt

The logic of above codes is explained in our technical report.

Error Detection and Correction

In this section, CorrectRealtime.py is used to detection and correct errors in video caption.

Location of Code

Server: orbit.cs.columbia.edu

Location of code:
/home/ruichi/Documents/OCR/ CorrectRealtime.py

The databases connection configurations are in:
/home/ruichi/Documents/OCR/Dictionary/DB_Config.txt

Important Functions

def DB_Info(path):

This function is to read DB information

def check_number(a):

Check whether a word is a number

def check_money(word):
Check whether a word is in a form of money

def check_time(word):
Check whether a word is in a form of time

def correct_number (sentence):
Correct errors in number format like: 30.000->30,000

def correct_number_without_comma(sentence):
Correct errors in number format like: 40 000-> 40,000

def creat_English_dict(path1):
Create English dictionary from:

/home/ruichi/Documents/OCR/Dictionary/wordsEn.txt

The English dictionary we use is from:
http://sourceforge.net/projects/wordlist/?source=typ_redirect

def creat_Mined_dict(path1):
Create special word dictionary from:
/home/ruichi/Documents/OCR/Dictionary/Mined.txt

def creat_Mapping(path1):
Create High-Frequency Error Mapping from:
/home/ruichi/Documents/OCR/Dictionary/special.txt

def creat_special_dict(path1):
Create a flexible dictionary contains manually chosen words from:
/home/ruichi/Documents/OCR/Dictionary/special.txt

def creat_transcript_dict(path1,id):
Create transcript dictionary

def creat_daily_dict(path1,day):
Create time based dictionary

def creat_event_dict(path1,eventid):
Create event dictionary

def Check_word(English_dict,transcript_dict,special_dict,word):
Check whether this is a wrong word. If the word is correct, return True.

def Check_Sentence(English_dict,transcript_dict, special_dict,sentence):
Check whether the sentence contains wrong word. If contains, return True.

def special_symbol(word):
This is used to recover errors with special symbol

def recover_special_symbot(word,index):
Given the word and special symbol index, recover the special symbol of it.

def plural(word):
Recover the original word from its several formats.

def Hierachical_Correction (error_mapping,error,before_error,after_error,transcript_dict,daily_dict,event_dict,special_dict,number,alpha, beta):

Use dictionaries with different level of priority to hierarchical correct a video caption error.

def correct_error(error,before_error,after_error,dictionary,number,alpha, beta):
Given any local dictionary, correct based on both Levenshtein Distance Metric and Language Model.

Potential Problem and Solution

Those codes above have been run for months. The only problem we faced is from databases. If databases fail, we need to reset the configuration table to unlock all program locks.