Table of Contents for Development History

TASK 1 2

CHECKPOINT 1 2
CHECKPOINT 2 2
CHECKPOINT 3 3

TASK 2 4

CHECKPOINT 1 4
CHECKPOINT 2 5
CHECKPOINT 3 6
CHECKPOINT 4 8
CHECKPOINT 5 9
CHECKPOINT 6 10

TASK 3 12

CHECKPOINT 1 12
CHECKPOINT 2 12
CHECKPOINT 3 13
CHECKPOINT 4 14

Task 1

Checkpoint 1

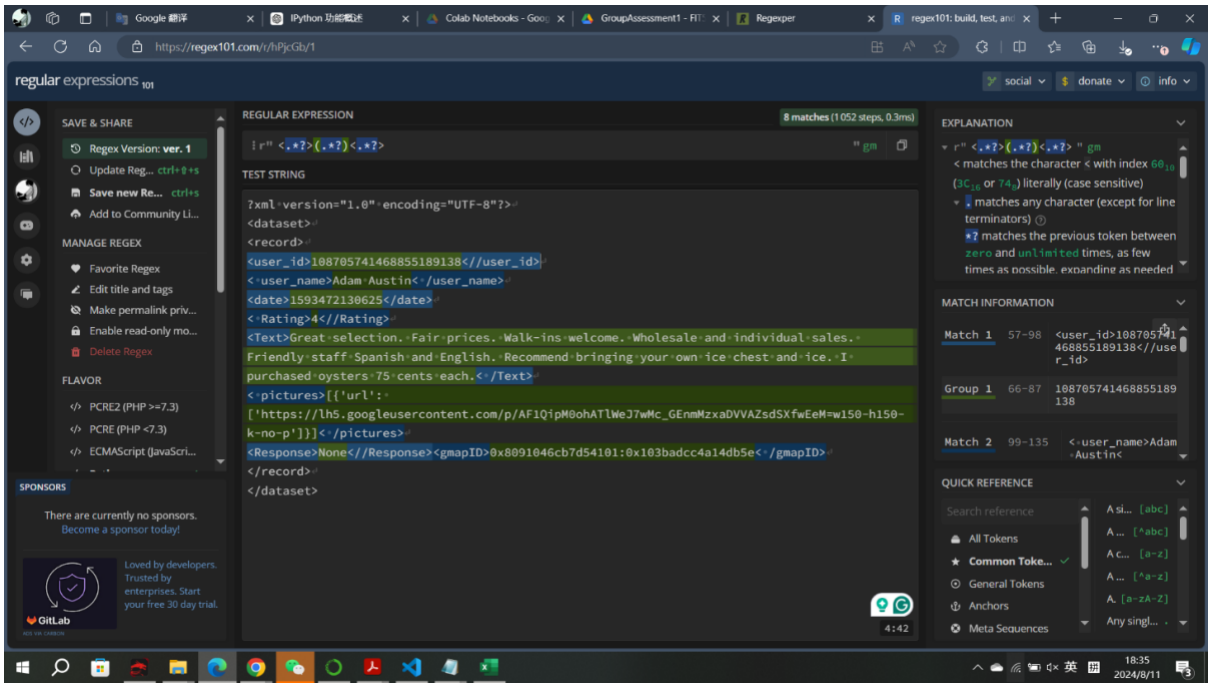
DateTime: 2024-08-11 18:35

Content: Try to use regex to extract the .txt data.

Task 1 State: task1_020.ipynb can load the .xlsx file.

Contribution: Zihan Yin

Proof:



Checkpoint 2

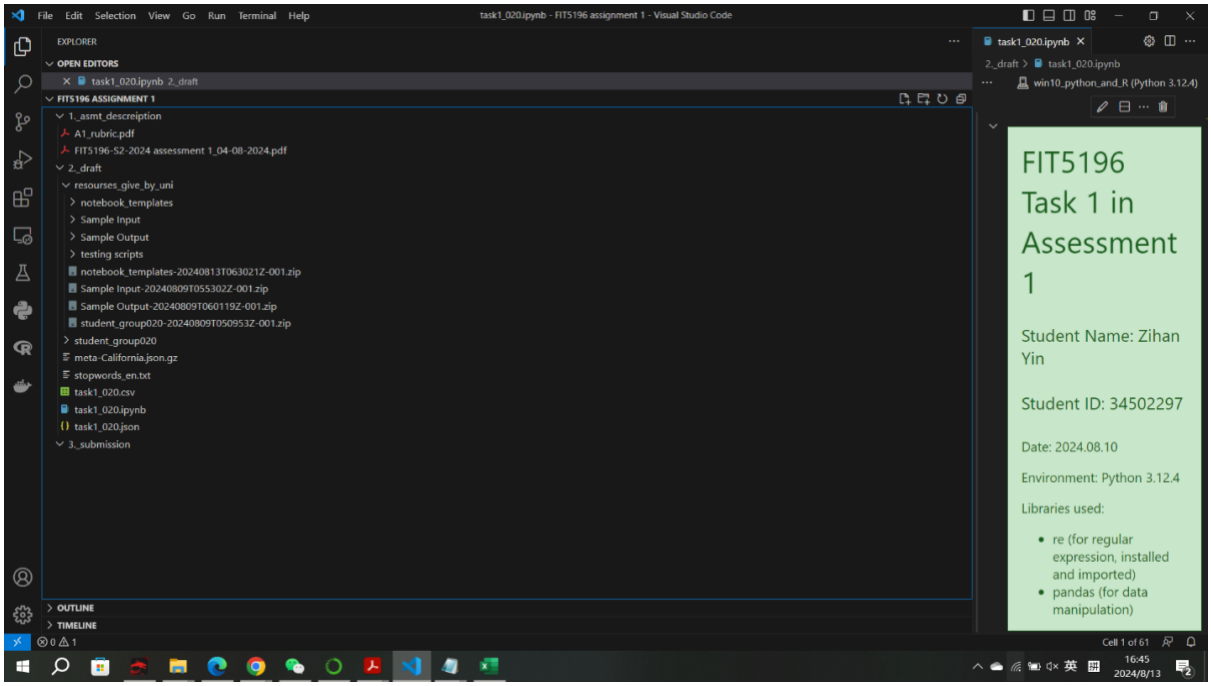
DateTime: 2024-08-13 16:45

Content: Finish all the codes of task 1.

Task 1 State: task1_020.ipynb can output the required .csv file & .json file, which are able to pass task1_test.py

Contribution: Zihan Yin

Proof:



Checkpoint 3

DateTime: 2024-08-16 4:28

Content:

Complete code comments, step explanations, and framework structures for task 1.

In particular:

1. Adjust the markdown format and write the interpretation

2. Adjust the code to reduce redundancy

3. Add appropriate code to show counterexamples

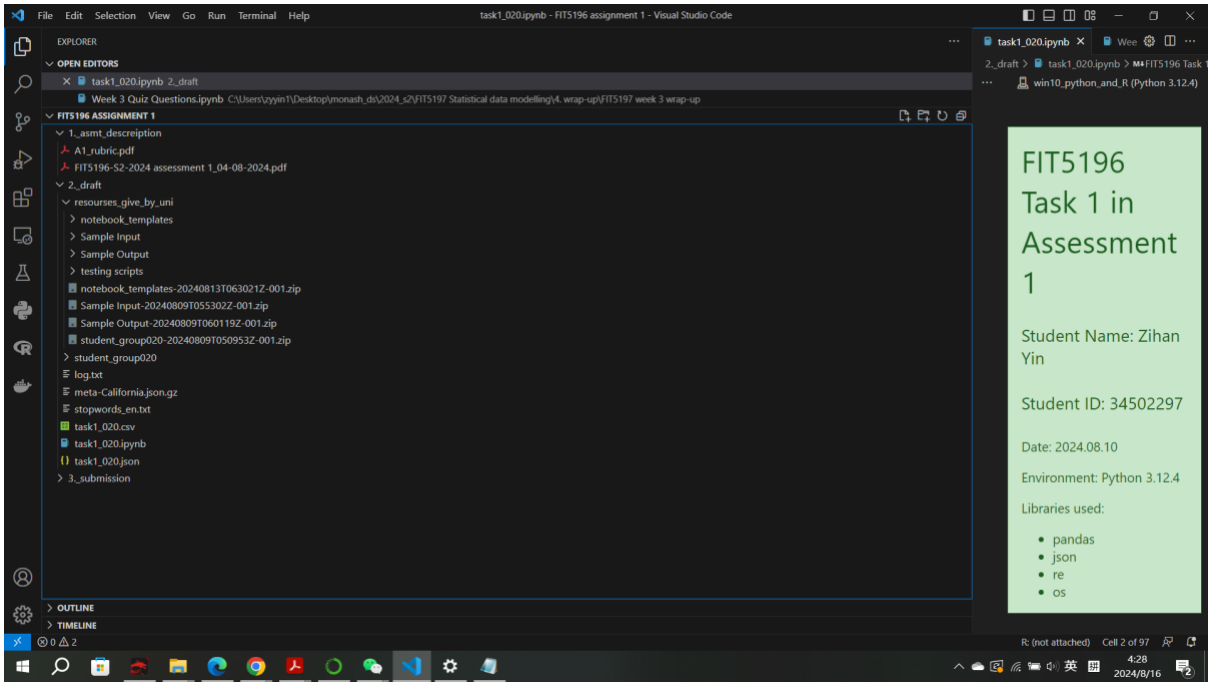
4. Write code comments

5. Check the accuracy repeatedly

State: all done with better readability.

Contribution: Zihan Yin

Proof:



Task 2

Checkpoint 1

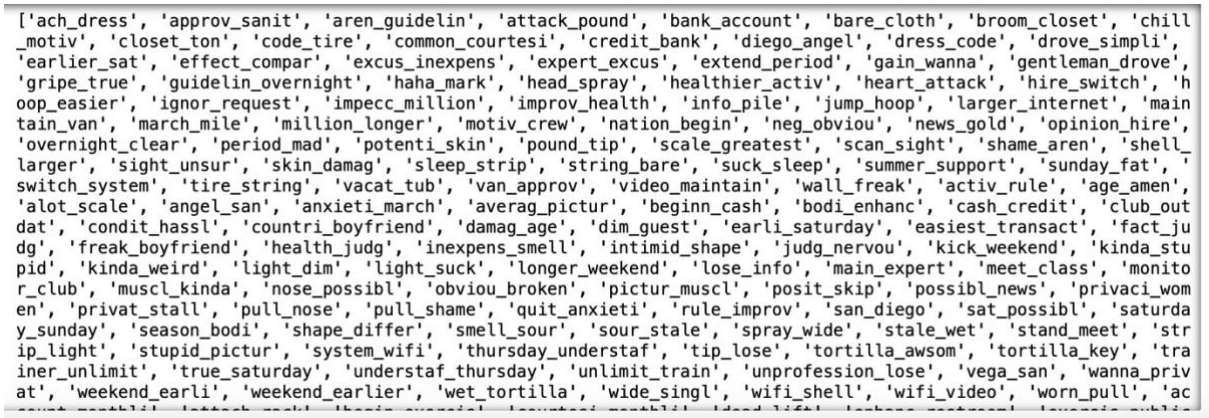
DateTime: 2024-08-24 18:12

Content: Extract meaningful bigrams.

Challenge: The output of bigrams after stemming does not meaningful anymore, words becomes like ‘earli_smell’ which is hard to be analysis. For words to be meaningful, the only way is to keep their original form. Thus, it is decided to not stem that bigrams. The challenge here is that how to stem unigram but not stem bigrams? The solution I apply here is to create two empty lists called ‘all_tokens’ and ‘unigram_list. One stores the token that only has been removed stop words from ‘stopwords_en.text’, while another one stores tokens that has been stemmed, removed stops as well as other threshold. In this case, bigrams and unigrams are operated in differnet lists. After finishing all the cleaning step, they are combined together and stored in a list called ‘final_vocabulary’.

Contribution: Ruiwen Chen

Proof:



```
['ach_dress', 'approv_sanit', 'aren_guidelin', 'attack_pound', 'bank_account', 'bare_cloth', 'broom_closet', 'chill
_motiv', 'closet_ton', 'code_tire', 'common_courtesi', 'credit_bank', 'diego_angel', 'dress_code', 'drove_simpli',
'earlier_sat', 'effect_compar', 'excus_inexpens', 'expert_excus', 'extend_period', 'gain_wanna', 'gentleman_drove',
'gripe_true', 'guidelin_overnight', 'haha_mark', 'head_spray', 'healthier_activ', 'heart_attack', 'hire_switch', 'h
oop_easier', 'ignor_request', 'impecc_million', 'improv_health', 'info_pile', 'jump_hoop', 'larger_internet', 'main
tain_van', 'march_mile', 'million_longer', 'motiv_crew', 'nation_begin', 'neg_obviou', 'news_gold', 'opinion_hire',
'overnight_clear', 'period_mad', 'potenti_skin', 'pound_tip', 'scale_greatest', 'scan_sight', 'shame_aren', 'shell_
larger', 'sight_unsur', 'skin_damag', 'sleep_strip', 'string_bare', 'suck_sleep', 'summer_support', 'sunday_fat', '
switch_system', 'tire_string', 'vacat_tub', 'van_approv', 'video_maintain', 'wall_freak', 'activ_rule', 'age_amen',
'alot_scale', 'angel_san', 'anxieti_march', 'averag_pictur', 'beginn_cash', 'bodi_enhanc', 'cash_credit', 'club_out
dat', 'condit_hassl', 'countri_boyfriend', 'damag_age', 'dim_guest', 'earli_saturday', 'easiest_transact', 'fact_ju
dg', 'freak_boyfriend', 'health_judg', 'inexpens_smell', 'intimid_shape', 'judg_nervou', 'kick_weekend', 'kinda_stu
pid', 'kinda_weird', 'light_dim', 'light_suck', 'longer_weekend', 'lose_info', 'main_expert', 'meet_class', 'monito
r_club', 'muscl_kinda', 'nose_possibl', 'obviou_broken', 'pictur_muscl', 'posit_skip', 'possibl_news', 'privaci_wom
en', 'privat_stall', 'pull_nose', 'pull_shame', 'quit_anxieti', 'rule_improv', 'san_diego', 'sat_possibl', 'saturda
y_sunday', 'season_bodi', 'shape_differ', 'smell_sour', 'sour_stale', 'spray_wide', 'stale_wet', 'stand_meet', 'str
ip_light', 'stupid_pictur', 'system_wifi', 'thursday_understaf', 'tip_lose', 'tortilla_awsom', 'tortilla_key', 'tra
iner_unlimit', 'true_saturday', 'understaf_thursday', 'unlimit_train', 'unprofession_lose', 'vega_san', 'wanna_priv
at', 'weekend_earli', 'weekend_earlier', 'wet_tortilla', 'wide_singl', 'wifi_shell', 'wifi_video', 'worn_pull', 'ac
```

```
['accommodating_providing', 'aches_pains', 'active_lifestyle', 'age_soledad', 'annoying_manning', 'answers_question
s', 'appeared_broom', 'aren_guidelines', 'attacks_stroke', 'automated_system', 'bandwidth_internet', 'bank_account'
, 'bar_lights', 'barely_clothes', 'beginner_short', 'beginning_exerciser', 'broom_closet', 'changing_stalls', 'char
ged_contract', 'childcare_class', 'classes_pools', 'club_outdated', 'code_tired', 'common_courtesy', 'completely_sh
ame', 'cook_aches', 'damage_premature', 'daunting_task', 'didn_single', 'diego_los', 'dress_code', 'dumbbells_lbs',
'earlier_sat', 'early_saturday', 'effective_comparing', 'elliptical_rowing', 'enforcevtgr_dress', 'equipped_wall',
'exercising_public', 'extended_periods', 'facilities_late', 'fact_judge', 'family_arthritis', 'fart_cook', 'fast_up
load', 'fees_haven', 'fox_news', 'francisco_denver', 'gains_end', 'garden_hose', 'giving_prayer', 'greatest_accommo
dating', 'gripe_true', 'grocery_store', 'groups_monopolizing', 'grunting_rule', 'haha_thought', 'hassles_judgement'
, 'head_rusted', 'healthier_active', 'heart_attacks', 'heavier_dumbells', 'hose_equipped', 'ignore_request', 'immea
diatly_freaked', 'include_privacy', 'info_letting', 'internet_busy', 'intervention_acknowledging', 'jump_hoops', 'l
ap_tool', 'large_number', 'larger_bandwidth', 'late_night', 'lb_pounds', 'left_travel', 'letting_piling', 'license_
year', 'lifestyle_grunting', 'lighting_sucks', 'lights_dim', 'los_angeles', 'lukewarm_partner', 'lunk_rules', 'main
tained_grocery', 'man_improve', 'manager_enforce', 'mark_ps', 'masks_refuse', 'massages_worst', 'meet_classes', 'me
mberships_technically', 'miles_town', 'minded_giving', 'morning_favorite', 'mothly_provide', 'nation_beginning', 'n
eeding_cleaned', 'negatives_obvious', 'nervous_exercising', 'ny_vegas', 'office_playing', 'online_daunting', 'op_fa
rt', 'order_meet', 'overnight_clears', 'overweight_man', 'pains_enforcevtgr', 'pf_memeberships', 'pictures_cartoon'
, 'piling_fees', 'pleasant_understanding', 'podunk_towns', 'policy_signs', 'pools_organized', 'potential_skin', 'pr
ayer_drove', 'preening_posing', 'premature_aging', 'product_excuse', 'professional_fat', 'provide_bank', 'radius_di
```

Checkpoint 2

DateTime: 2024-08-25 16:00

Challenge: Not sure how to check whether the 200 bigrams are in the text. Initially, i am thinking to use a loop to loop through both bigrams and text, if they match, they append in a list. This one works, but it might be less effective because it need to loop a lot. Then, I am thinking if `ngram` works or not, it turns out it is the same output as the loop one, then, I choose to with this because it is more concise.

Contribution: Ruiwen Chen

Proof:

```
valid_bigrams = []
for bigram in bigram_vocab:
    word1, word2 = bigram.split('_')

    # Check if word1 and word2 appear consecutively in the original tokenized text
    for i in range(len(all_tokens) - 1):
        if all_tokens[i] == word1 and all_tokens[i + 1] == word2:
            valid_bigrams.append(bigram)
            break  # Once found, no need to check further for this bigram

print(valid_bigrams)
print(len(valid_bigrams))
```

```
['accommodating_providing', 'aches_pains', 'active_lifestyle', 'annoying_manning', 'answers_questions', 'appeared_b
room', 'aren_guidelines', 'automated_system', 'bank_account', 'barely_clothes', 'beginning_exerciser', 'broom_close
t', 'changing_stalls', 'charged_contract', 'childcare_class', 'classes_pools', 'club_outdated', 'code_tired', 'comm
on_courtesy', 'completely_shame', 'damage_premature', 'daunting_task', 'diego_los', 'dress_code', 'dumbbells_lbs',
'earlier_sat', 'early_saturday', 'effective_comparing', 'elliptical_rowing', 'equipped_wall', 'exercising_public',
'extended_periods', 'fact_judge', 'family_arthritis', 'fees_haven', 'greatest_accommodating', 'gripe_true', 'groups
_monopolizing', 'grunting_rule', 'hassles_judgement', 'head_rusted', 'healthier_active', 'heart_attacks', 'ignore_r
equest', 'immeadiatly_freaked', 'include_privacy', 'info_letting', 'intervention_acknowledging', 'jump_hoops', 'let
ting_piling', 'lifestyle_grunting', 'lighting_sucks', 'lights_dim', 'los_angeles', 'maintained_grocery', 'manager_e
nforce', 'masks_refuse', 'meet_classes', 'memberships_technically', 'minded_giving', 'morning_favorite', 'mothly_pr
ovide', 'nation_beginning', 'needing_cleaned', 'negatives_obvious', 'office_playing', 'online_daunting', 'overnight
_clears', 'piling_fees', 'policy_signs', 'pools_organized', 'potential_skin', 'preening_posing', 'premature_aging',
'professional_fat', 'provide_bank', 'rates_equipments', 'reasonable_operating', 'rock_attentive', 'rusted_sprayed',
'saturday_morning', 'saturdays_sundays', 'shell_larger', 'simply_amaze', 'skin_damage', 'sleep_strip', 'spa_works',
'string_tops', 'stupid_pictures', 'summer_quality', 'switch_automated', 'task_cancelled', 'theyve_stand', 'thoughtf
ully_answers', 'tired_string', 'tops_clotgre', 'towns_understaffed', 'travelling_rates', 'unsure_mens', 'walking_mi
les', 'wall_immeadiatly', 'wearing_masks', 'wondering_patrons', 'worked_coming', 'affordable_extremely', 'angeles_s
an', 'asked_minded', 'attach_rack', 'barbell_attach', 'barbell_bench', 'barbells_lead', 'bathrooms_millions', 'bath
rooms_needing', 'bed_ten', 'benches_barbell', 'body_builders', 'body_enhancement', 'boutique_style', 'cancelled_eas
iest', 'cash_credit', 'catches_kidding', 'chair_couple', 'closed_thursday', 'clothes_traveling', 'committments_catc
hes', 'contract_cancelling', 'country_boyfriend', 'crowded_feeling', 'crowded_wondering', 'customers_services', 'cu
stomers_wearing', 'days_notice', 'easiest_transaction', 'employee_sight', 'excellent_condition', 'experts_knowledge
able', 'extremely_annoying', 'freaked_boyfriend', 'guys_barely', 'guys_muscles', 'including_manager', 'including_sm
ith']
142
```

```
token_bigrams = list(ngrams(all_tokens, 2))

    # Filter out bigrams that do not exist in the original tokenized text
valid_bigrams = []
for bigram in bigram_vocab:
    word1, word2 = bigram.split('_')

        # Check if the bigram is in the list of generated bigrams
    if (word1, word2) in token_bigrams:
        valid_bigrams.append(bigram)

    # Update the final_gmap_dict with only valid bigrams
    #final_gmap_dict[gmap_id] = unigram_vocab + valid_bigrams

print(valid_bigrams)
print(len(valid_bigrams))
```

```
_monopolizing', 'grunting_rule', 'hassles_judgement', 'head_rusted', 'healthier_active', 'heart_attacks', 'ignore_r
equest', 'immeadiatly_freaked', 'include_privacy', 'info_letting', 'intervention_acknowledging', 'jump_hoops', 'let
ting_piling', 'lifestyle_grunting', 'lighting_sucks', 'lights_dim', 'los_angeles', 'maintained_grocery', 'manager_e
nforce', 'masks_refuse', 'meet_classes', 'memberships_technically', 'minded_giving', 'morning_favorite', 'mothly_pr
ovide', 'nation_beginning', 'needing_cleaned', 'negatives_obvious', 'office_playing', 'online_daunting', 'overnight
_clears', 'piling_fees', 'policy_signs', 'pools_organized', 'potential_skin', 'preening_posing', 'premature_aging',
'professional_fat', 'provide_bank', 'rates_equipments', 'reasonable_operating', 'rock_attentive', 'rusted_sprayed',
'saturday_morning', 'saturdays_sundays', 'shell_larger', 'simply_amaze', 'skin_damage', 'sleep_strip', 'spa_works',
'string_tops', 'stupid_pictures', 'summer_quality', 'switch_automated', 'task_cancelled', 'theyve_stand', 'thoughtf
ully_answers', 'tired_string', 'tops_clotgre', 'towns_understaffed', 'travelling_rates', 'unsure_mens', 'walking_mi
les', 'wall_immeadiatly', 'wearing_masks', 'wondering_patrons', 'worked_coming', 'affordable_extremely', 'angeles_s
an', 'asked_minded', 'attach_rack', 'barbell_attach', 'barbell_bench', 'barbells_lead', 'bathrooms_millions', 'bath
rooms_needing', 'bed_ten', 'benches_barbell', 'body_builders', 'body_enhancement', 'boutique_style', 'cancelled_eas
iest', 'cash_credit', 'catches_kidding', 'chair_couple', 'closed_thursday', 'clothes_traveling', 'committments_catc
hes', 'contract_cancelling', 'country_boyfriend', 'crowded_feeling', 'crowded_wondering', 'customers_services', 'cu
stomers_wearing', 'days_notice', 'easiest_transaction', 'employee_sight', 'excellent_condition', 'experts_knowledge
able', 'extremely_annoying', 'freaked_boyfriend', 'guys_barely', 'guys_muscles', 'including_manager', 'including_sm
ith']
142
```

Checkpoint 3

DateTime: 2024-08-25 18:12

Challenge: When checking if the 200 bigram is in the text, the challenge is that which text should we use, the original text or the one after remove stopwords? The output gives 50 words out of 200 if we use original text, it gives 142 if we use the one after remove stopwords. From the quality of output, the 50 words seems more meaningful and we can abstract more information by only look at the bigrams. For the 142, some of them make sense and some of them is not that clear. Zihan think that we need to use the one after stemming and cleaning, I think we need to go with original text. Thus, we had some discussion on that. Now, we decide to use the one after remove stopwords because it not only includes the same 50 from original text, but also gives more words to analysis. Sometime more can be better to analysis.

Contribution: Ruiwen Chen

Proof:

```
# Iterate through tokenizer_dict to get original text
for gmap_id, da in tokenizer_dict.items():
    original_text_tokens = []

    # Gather all tokens from the tokenizer_dict for this gmap_id
    for date, tokens in da.items():
           original_text_tokens.extend(tokens)

    # Generate bigrams from the original tokens
    token_bigrams = list(ngrams(original_text_tokens, 2))

        # Filter out bigrams that do not exist in the original tokenized text
    valid_bigrams = []
    for bigram in bigram_vocab:
        word1, word2 = bigram.split('_')

            # Check if the bigram is in the list of generated bigrams
        if (word1, word2) in token_bigrams:
            valid_bigrams.append(bigram)

    # Update the final_gmap_dict with only valid bigrams
    #final_gmap_dict[gmap_id] = unigram_vocab + valid_bigrams
#print(original_text_tokens)
print(valid_bigrams)
print(len(valid_bigrams))
```

```
['active_lifestyle', 'answers_questions', 'automated_system', 'bank_account', 'beginning_exerciser', 'broom_closet'
, 'changing_stalls', 'classes_pools', 'code_tired', 'common_courtesy', 'daunting_task', 'dress_code', 'early_saturd
ay', 'effective_comparing', 'extended_periods', 'groups_monopolizing', 'grunting_rule', 'healthier_active', 'heart_
attacks', 'immeadiatly_freaked', 'include_privacy', 'lighting_sucks', 'los_angeles', 'overnight_clears', 'pools_org
anized', 'premature_aging', 'reasonable_operating', 'saturday_morning', 'simply_amaze', 'skin_damage', 'spa_works',
'string_tops', 'summer_quality', 'thoughtfully_answers', 'barbell_bench', 'bathrooms_millions', 'bathrooms_needing'
, 'body_builders', 'body_enhancement', 'boutique_style', 'closed_thursday', 'committments_catches', 'crowded_feelin
g', 'crowded_wondering', 'customers_services', 'days_notice', 'easiest_transaction', 'excellent_condition', 'extrem
ely_annoying', 'including_manager']
50
```

```
token_bigrams = list(ngrams(all_tokens, 2))

    # Filter out bigrams that do not exist in the original tokenized text
valid_bigrams = []
for bigram in bigram_vocab:
    word1, word2 = bigram.split('_')

        # Check if the bigram is in the list of generated bigrams
    if (word1, word2) in token_bigrams:
        valid_bigrams.append(bigram)

    # Update the final_gmap_dict with only valid bigrams
    #final_gmap_dict[gmap_id] = unigram_vocab + valid_bigrams

print(valid_bigrams)
print(len(valid_bigrams))
```

```
_monopolizing', 'grunting_rule', 'hassles_judgement', 'head_rusted', 'healthier_active', 'heart_attacks', 'ignore_r
equest', 'immeadiatly_freaked', 'include_privacy', 'info_letting', 'intervention_acknowledging', 'jump_hoops', 'let
ting_piling', 'lifestyle_grunting', 'lighting_sucks', 'lights_dim', 'los_angeles', 'maintained_grocery', 'manager_e
nforce', 'masks_refuse', 'meet_classes', 'memberships_technically', 'minded_giving', 'morning_favorite', 'mothly_pr
ovide', 'nation_beginning', 'needing_cleaned', 'negatives_obvious', 'office_playing', 'online_daunting', 'overnight
_clears', 'piling_fees', 'policy_signs', 'pools_organized', 'potential_skin', 'preening_posing', 'premature_aging',
'professional_fat', 'provide_bank', 'rates_equipments', 'reasonable_operating', 'rock_attentive', 'rusted_sprayed',
'saturday_morning', 'saturdays_sundays', 'shell_larger', 'simply_amaze', 'skin_damage', 'sleep_strip', 'spa_works',
'string_tops', 'stupid_pictures', 'summer_quality', 'switch_automated', 'task_cancelled', 'theyve_stand', 'thoughtf
ully_answers', 'tired_string', 'tops_clotgre', 'towns_understaffed', 'travelling_rates', 'unsure_mens', 'walking_mi
les', 'wall_immeadiatly', 'wearing_masks', 'wondering_patrons', 'worked_coming', 'affordable_extremely', 'angeles_s
an', 'asked_minded', 'attach_rack', 'barbell_attach', 'barbell_bench', 'barbells_lead', 'bathrooms_millions', 'bath
rooms_needing', 'bed_ten', 'benches_barbell', 'body_builders', 'body_enhancement', 'boutique_style', 'cancelled_eas
iest', 'cash_credit', 'catches_kidding', 'chair_couple', 'closed_thursday', 'clothes_traveling', 'committments_catc
hes', 'contract_cancelling', 'country_boyfriend', 'crowded_feeling', 'crowded_wondering', 'customers_services', 'cu
stomers_wearing', 'days_notice', 'easiest_transaction', 'employee_sight', 'excellent_condition', 'experts_knowledge
able', 'extremely_annoying', 'freaked_boyfriend', 'guys_barely', 'guys_muscles', 'including_manager', 'including_sm
ith']
142
```

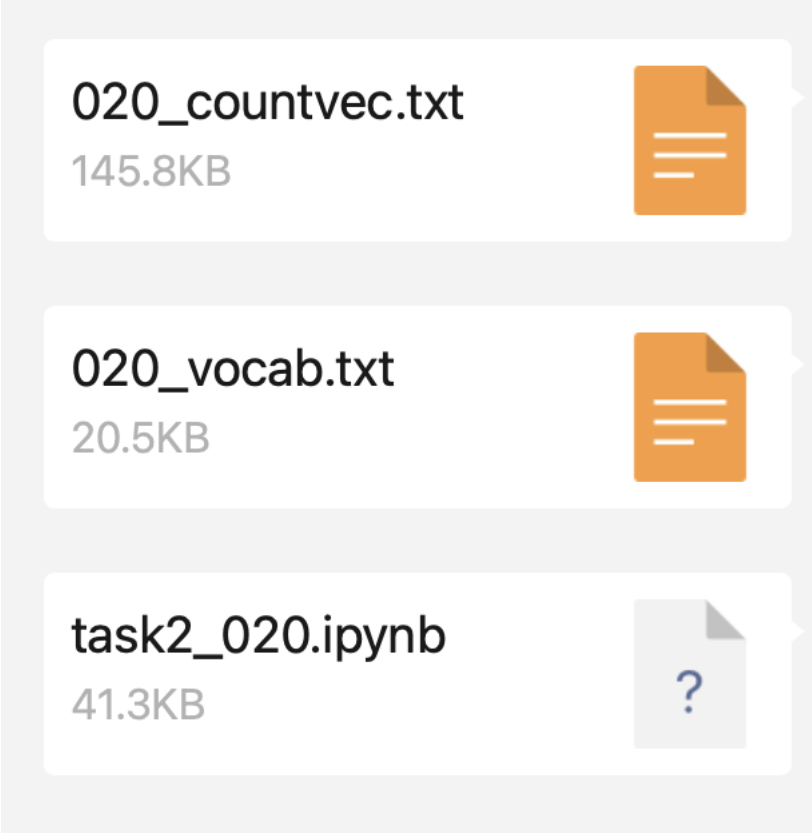
Checkpoint 4

DateTime: 2024-08-27 18:12

Content: Finish all the codes of task 2.

Task 2 State: task2_020.ipynb can output the required .txt files, which are able to pass task2_test.py. task2_020.ipynb notebook has been handed over to Zihan Yin for some additional adjustments.

Contribution: Ruiwen Chen

Proof: 

Checkpoint 5

DateTime: 2024-08-29 6:41

Content: Make minor changes to task2_020.ipynb.

Considere that the output of task 2 is wrong.

Find out some issues existing in task 2

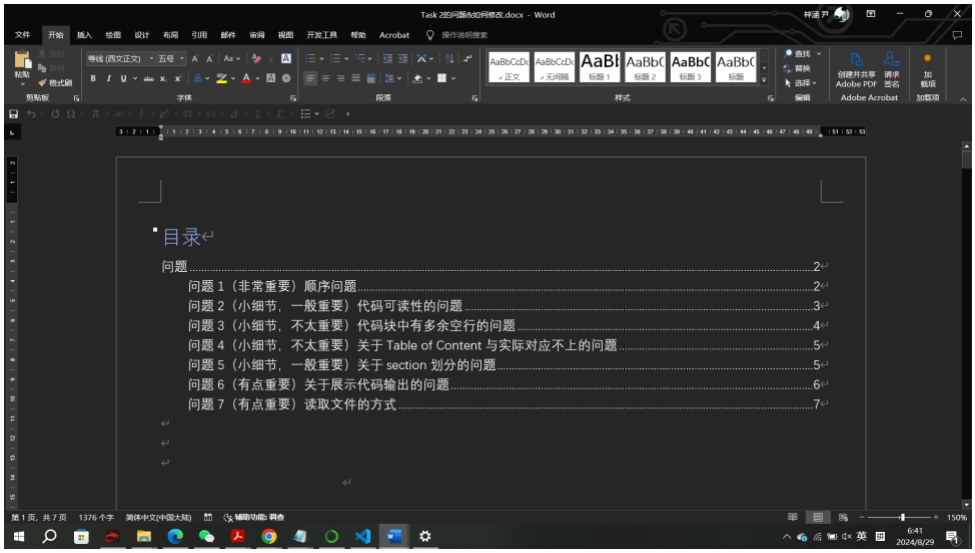
Create a word document to explain the issues and details that needed to be modified

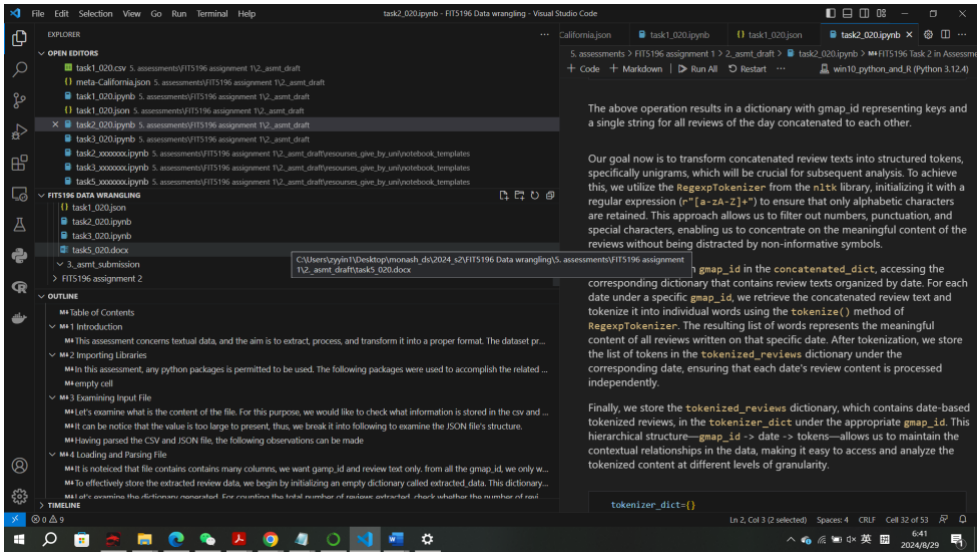
State: Hand over the word document explaining the issues together with the task2_020.ipynb file to Ruiwen Chen.

Contribution: Zihan Yin

Proof:

<Here's a screenshot of the word document explaining the issues. Please ignore the language used in this document to describe the issues>





Checkpoint 6

DateTime: 2024-08-29 7:00

Content: Frequency list are all 1.

Challenge: The frequency in countvec.txt are all 1, which is meaningless for analysis and there must be something wrong. I checked the code again and identified that I used set() when create ‘all_tokens’ and ‘unigram_list. Set() function will filter out the repetition automatically, result in no repetition. The solution I apply to this is to change the set() to lists [].

Contribution: Ruiwen Chen

Proof:

```
0x54cb977215fdb773:0xa8189ecf8ad14096, 105:1, 223:1, 645:1, 654:1, 722:1, 962:1, 1000:1, 1051:1,
1144:1, 1180:1, 1701:1, 1792:1, 2178:1, 2196:1, 2377:1, 2382:1, 2567:1, 2628:1, 2663:1, 2716:1,
2744:1, 2891:1, 2968:1, 3274:1, 3315:1, 3518:1, 3656:1, 3666:1, 3692:1, 3818:1, 3926:1, 4341:1,
4351:1, 4429:1, 4562:1, 4568:1, 4619:1, 4652:1, 4701:1, 4749:1, 4782:1, 4790:1, 4886:1, 4908:1,
4938:1, 5017:1, 5020:1, 5223:1, 5331:1, 5364:1, 5514:1, 5535:1, 5585:1, 5630:1, 5751:1, 5860:1,
6001:1, 6108:1, 6148:1, 6171:1, 6445:1, 6501:1, 6537:1, 6632:1, 6673:1, 6705:1, 6725:1, 6758:1,
6829:1, 6867:1, 6969:1, 7009:1, 7032:1, 7144:1, 7458:1, 7490:1, 7615:1, 7638:1, 7651:1, 7668:1,
7671:1, 7869:1, 7875:1, 7973:1, 8102:1, 8112:1, 8280:1, 8325:1, 8346:1, 8362:1, 8424:1, 8459:1,
8547:1, 8592:1, 8602:1, 8741:1, 8996:1, 9068:1, 9080:1, 9110:1, 9134:1, 9228:1, 9368:1, 9400:1,
9418:1, 9596:1, 9720:1, 9801:1, 9819:1, 9826:1, 9896:1, 9997:1, 10104:1, 10172:1, 10299:1, 10305:1,
10386:1, 10395:1, 10431:1, 10556:1, 10571:1, 10643:1, 10711:1, 10748:1, 10763:1, 10801:1, 10806:1,
10819:1, 10832:1, 10857:1, 10872:1, 10901:1, 11022:1, 11108:1, 11159:1, 11177:1, 11191:1, 11213:1,
11280:1, 11332:1, 11349:1, 11369:1, 11401:1, 11405:1, 11425:1, 11449:1, 11459:1, 11514:1, 11559:1,
11587:1, 11776:1, 11789:1, 11812:1, 11817:1, 11848:1, 11853:1, 11874:1, 11883:1, 11899:1, 11963:1,
11979:1, 12031:1, 12043:1, 12080:1, 12085:1, 12092:1, 12108:1, 12119:1, 12141:1, 12153:1, 12158:1,
12159:1, 12196:1, 12219:1, 12259:1, 12316:1, 12326:1, 12333:1, 12339:1, 12367:1, 12374:1, 12439:1,
12451:1, 12462:1, 12466:1, 12478:1, 12480:1, 12483:1, 12496:1, 12573:1, 12622:1, 12714:1, 12717:1,
12741:1, 12749:1, 12785:1, 12799:1, 12829:1, 12847:1, 12881:1
0x54d2944ea7c764e9:0x3495e6456d6df90a, 57:1, 132:1, 283:1, 286:1, 357:1, 503:1, 674:1, 898:1, 1009:1,
1054:1, 1217:1, 1299:1, 1428:1, 1781:1, 1972:1, 2115:1, 2439:1, 2591:1, 2609:1, 2648:1, 2709:1,
2764:1, 3063:1, 3205:1, 3216:1, 3476:1, 3532:1, 3563:1, 3636:1, 3679:1, 3723:1, 3781:1, 3783:1,
4443:1, 4451:1, 4471:1, 4494:1, 4518:1, 4538:1, 5027:1, 5085:1, 5274:1, 5295:1, 5298:1, 5315:1,
5334:1, 5421:1, 5631:1, 5759:1, 5956:1, 6060:1, 6262:1, 6390:1, 6764:1, 6814:1, 6831:1, 6854:1,
6871:1, 6873:1, 7117:1, 7124:1, 7143:1, 7177:1, 7194:1, 7217:1, 7314:1, 7478:1, 7579:1, 7595:1,
7811:1, 7884:1, 8090:1, 8092:1, 8223:1, 8260:1, 8289:1, 8394:1, 8402:1, 8492:1, 8740:1, 8938:1,
8943:1, 8979:1, 9152:1, 9323:1, 9386:1, 9414:1, 9465:1, 9528:1, 9551:1, 9558:1, 9580:1, 9600:1,
9657:1, 9663:1, 9690:1, 9701:1, 9716:1, 9922:1, 9963:1, 10051:1, 10093:1, 10107:1, 10207:1, 10270:1,
10325:1, 10354:1, 10420:1, 10446:1, 10512:1, 10531:1, 10551:1, 10601:1, 10634:1, 10706:1, 10735:1,
10742:1, 10776:1, 10797:1, 10823:1, 10876:1, 11055:1, 11056:1, 11096:1, 11124:1, 11129:1, 11165:1,
11178:1, 11190:1, 11266:1, 11314:1, 11332:1, 11355:1, 11389:1, 11426:1, 11500:1, 11587:1, 11661:1,
```

After adjustment, frequency are not all 1.

```
0x54cb977215fdb773:0xa8189ecf8ad14096, 54:1, 122:1, 344:1, 401:1, 533:1, 540:6, 594:1, 1075:5,
1159:5, 1161:1, 1163:1, 1254:15, 1305:10, 1330:1, 1423:1, 1468:1, 1630:1, 1646:1, 1758:6, 1847:1,
1852:1, 1856:1, 1863:1, 1937:1, 1993:1, 2195:25, 2203:1, 2232:5, 2324:1, 2345:1, 2417:1, 2430:12,
2503:1, 2524:1, 2560:5, 2669:1, 2751:1, 2814:8, 2835:1, 2868:1, 2894:1, 2951:1, 3027:1, 3167:1,
3168:1, 3191:5, 3214:1, 3395:1, 3405:4, 3414:1, 3418:1, 3476:1, 3519:1, 3556:1, 3614:5, 3780:1,
3979:1, 3997:1, 4102:13, 4135:1, 4273:1, 4347:1, 4348:1, 4442:1, 4455:1, 4614:1, 4627:1, 4706:1,
4707:1, 4712:1, 4828:1, 4925:7, 5472:5, 5505:1, 5820:1, 5884:11, 5901:1, 5959:5, 6174:1, 6353:14,
6414:1, 6420:1, 6440:1, 6611:1, 6722:9, 6765:1, 6795:1, 6834:1, 6861:1, 6928:1, 7071:1, 7227:1,
7248:1, 7385:1, 7471:1, 7485:5, 7503:6, 7538:15, 7656:1, 7751:1, 7961:46, 8011:1, 8012:1, 8051:1,
8088:5, 8098:67, 8186:1, 8191:1, 8276:1, 8285:1, 8347:1, 8366:1, 8432:1, 8490:1, 8493:1, 8494:1,
8764:1, 8813:1, 8831:1, 8861:14, 8988:1, 9056:1, 9057:6, 9099:5, 9107:1, 9354:1, 9808:1, 9827:1,
9943:1, 9986:1, 10087:1, 10127:6, 10251:4, 10417:1, 10576:1, 10588:1, 10607:1, 10652:1, 10719:1,
10733:1, 10754:9, 10764:1, 10781:1, 10844:1, 10863:4, 10900:1, 10923:7, 10961:1, 10967:1, 11142:8,
11152:1, 11247:1, 11286:4, 11297:1, 11474:1, 11507:1, 11562:1, 11596:5, 11616:1, 11663:1, 11673:5,
11896:1, 11956:6, 12196:1, 12247:1, 12325:1, 12370:1, 12411:1, 12417:1, 12517:1, 12672:1, 12683:1,
12724:1, 12729:1, 12739:1, 12803:1, 12806:1, 12808:1, 12870:10, 12955:1, 13144:28, 13163:1, 13186:1,
13192:1, 13197:1, 13263:1, 13272:1, 13273:1, 13404:6, 13464:1, 13542:1, 13605:5, 13621:11, 14068:1,
14173:1, 14276:6, 14298:1, 14369:1, 14627:7, 14634:1, 14709:1, 14787:1, 14814:1, 14946:1, 14955:1,
15054:1, 15075:1, 15133:1, 15137:1, 15246:1, 15410:4, 15424:1, 15431:36, 15523:1, 15669:4, 15883:5,
15901:1, 15973:1, 16089:1, 16110:1, 16218:4, 16273:7, 16417:5, 16495:11, 16528:1, 16600:1, 16615:1,
16669:1, 16978:4, 16994:4, 17190:1, 17199:1, 17289:6, 17323:1, 17357:1, 17502:7, 17581:1, 17591:1,
17605:1, 18007:1, 18101:1, 18113:1, 18141:5, 18150:1, 18164:8, 18262:1, 18312:1, 18337:1, 18357:1,
18404:1, 18514:1, 18521:1, 18567:1, 18574:1, 18577:1, 18592:1, 18608:6, 18664:1, 18681:1, 18686:6,
18690:1, 18692:1, 18746:1
0x54d2944ea7c764e9:0x3495e6456d6df90a, 23:1, 30:1, 70:1, 155:1, 205:1, 481:1, 540:4, 613:1, 657:1,
715:1, 977:1, 1043:1, 1197:1, 1254:3, 1266:1, 1305:3, 1334:1, 1525:1, 1594:1, 1597:1, 1733:1, 1758:2,
1787:1, 1836:1, 2232:2, 2253:1, 2264:8, 2265:1, 2275:1, 2296:4, 2595:1, 2709:1, 2710:1, 2715:3,
2716:1, 2895:5, 2956:1, 3068:1, 3133:1, 3268:1, 3334:1, 3560:1, 3592:1, 3614:36, 3626:1, 3760:1,
3762:1, 3767:1, 3778:1, 3815:1, 3884:1, 4080:1, 4228:1, 4291:1, 4427:1, 4537:6, 4565:1, 4572:8,
4662:1, 4746:1, 4925:1, 5101:1, 5139:1, 5147:1, 5288:1, 5472:2, 5486:1, 5515:1, 5630:17, 5656:1,
```

Task 3

Checkpoint 1

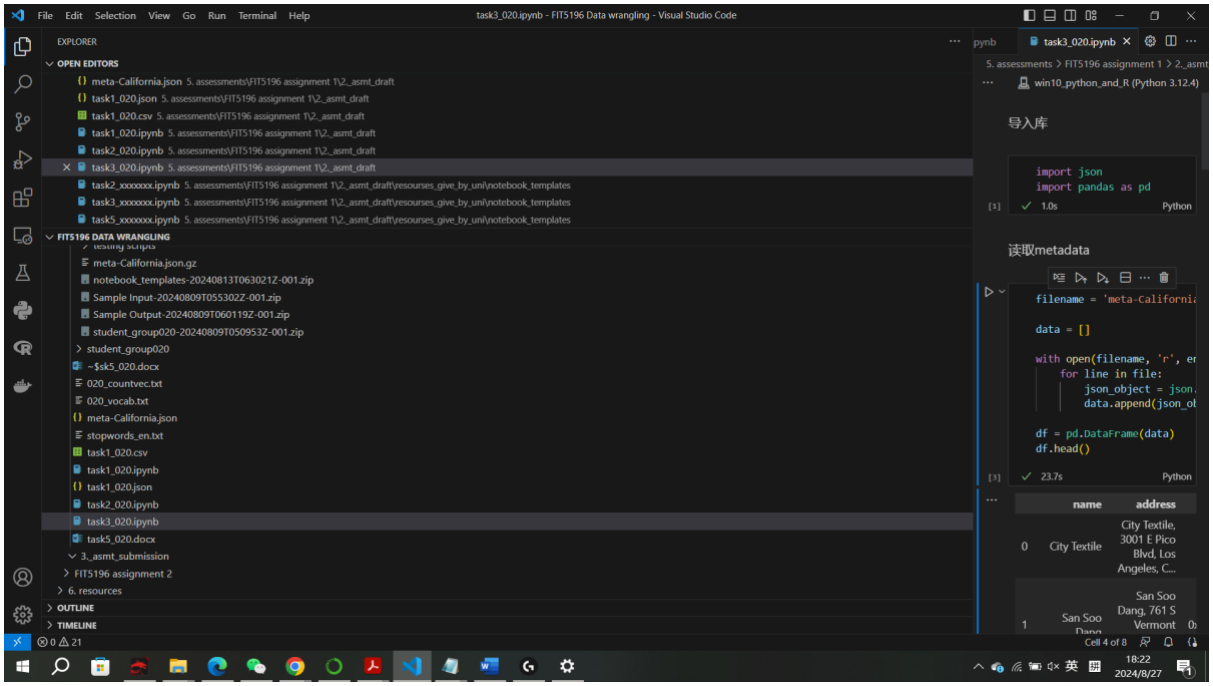
DateTime: 2024-08-27 18:12

Content: None

State: task3_020.ipynb is just started, and loading the metadata.

Contribution: Zihan Yin

Proof:



Checkpoint 2

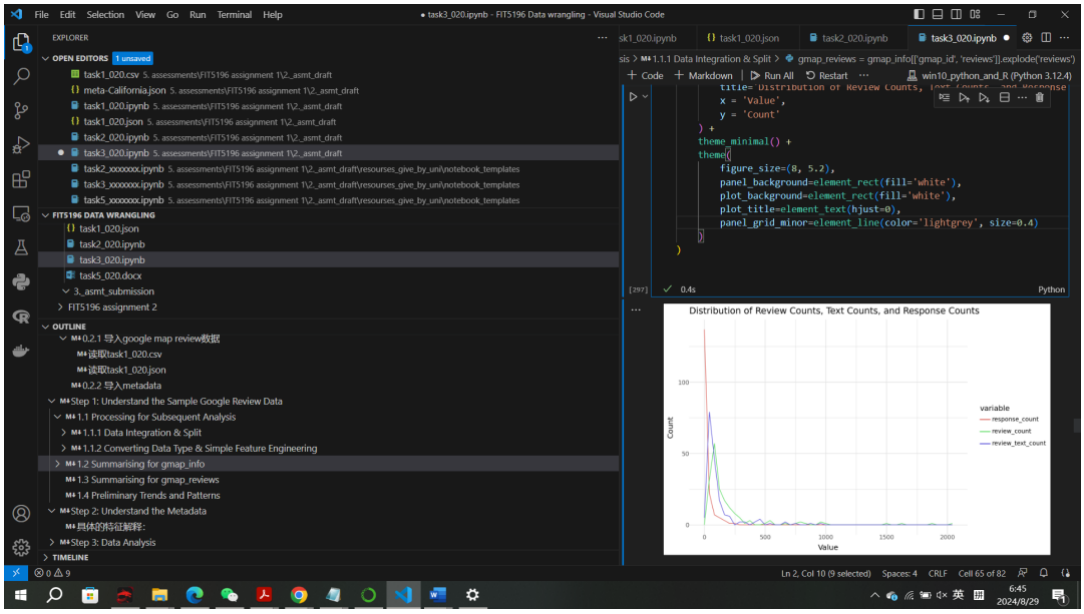
DateTime: 2024-08-29 18:45

Content: None

Task 3 State:

The Step 1 Understanding the sample google review data is halfway done. Step 2 and step 3 have already conceived some insights.

Contribution: Zihan Yin

Proof: 

Checkpoint 3

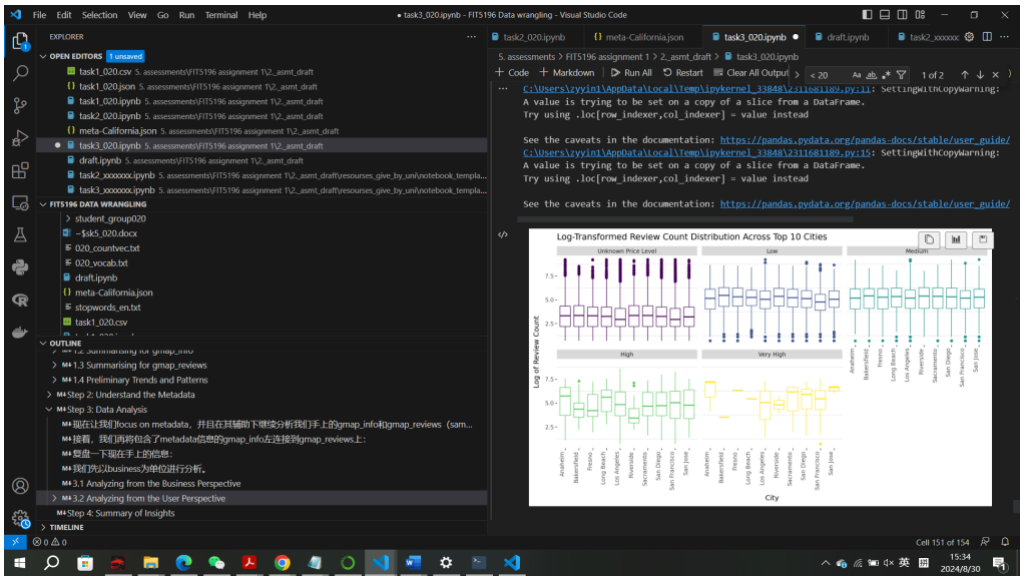
DateTime: 2024-08-30 15:34

Content: None

Task 3 State:

Step 1 & 2 has been finished， and step 3 is half-way done.

Contribution: Zihan Yin

Proof: 

Checkpoint 4

DateTime: 2024-08-30 18:25

Content: None

Task 3 State:

All done.

Contribution: Zihan Yin

Proof: