

Readme Document

The dataset is Yelp academic dataset downloaded from Yelp website:

https://www.yelp.com/dataset_challenge/dataset

It contains the following files relevant to our project:

A. yelp_academic_dataset_business.json

a. Contains a list of restaurants, and their categories

business_id	full_address	hours	categories
"5UmKMjUEUNdYWqANhGckJw"	"4734 Lebanon Church Rd\nDravosburg, PA 15034"	"{\n\"Friday\": {\n\"close\": \"21:00\", \"open\": \"11:00\"}\n}"	"{\n\"open\": true, \"categories\": [\n\"Nightlife\", \"city\": \"M\"

b. Sample record format:

{

"business_id": "5UmKMjUEUNdYWqANhGckJw",

"full_address": "4734 Lebanon Church Rd\nDravosburg, PA 15034",

"hours": {

"Friday": {

"close": "21:00",

"open": "11:00"

},

"Tuesday": {

"close": "21:00",

"open": "11:00"

},

"Thursday": {

"close": "21:00",

"open": "11:00"

},

```
    "Wednesday": {
      "close": "21:00",
      "open": "11:00"
    },
    "Monday": {
      "close": "21:00",
      "open": "11:00"
    }
  },
  "open": true,
  "categories": [
    "Fast Food",
    "Restaurants"
  ],
  "city": "Dravosburg",
  "review_count": 7,
  "name": "Mr Hoagie",
  "neighborhoods": [],
  "longitude": -79.9007057,
  "state": "PA",
  "stars": 3.5,
  "latitude": 40.3543266,
  "attributes": {
    "Take-out": true,
    "Drive-Thru": false,
    "Good For": {
      "dessert": false,
      "latenight": false,
      "lunch": false,
      "dinner": false,
      "brunch": false,
      "breakfast": false
    },
    "Caters": false,
    "Noise Level": "average",
    "Takes Reservations": false,
    "Delivery": false,
    "Ambience": {
      "romantic": false,
      "intimate": false,
      "classy": false,
      "hipster": false,
      "divey": false,

```

```
        "touristy": false,  
        "trendy": false,  
        "upscale": false,  
        "casual": false  
    },  
    "Parking": {  
        "garage": false,  
        "street": false,  
        "validated": false,  
        "lot": false,  
        "valet": false  
    },  
    "Has TV": false,  
    "Outdoor Seating": false,  
    "Attire": "casual",  
    "Alcohol": "none",  
    "Waiter Service": false,  
    "Accepts Credit Cards": true,  
    "Good for Kids": true,  
    "Good For Groups": true,  
    "Price Range": 1  
},  
"type": "business"  
}
```

- B. yelp_academic_dataset_review.json
 - a. Contains the reviews of the restaurants

	relevant_business_ids_and_reviews_v2_combined.json	relevant_business_ids_and_categories_only_one.json	yelp_academic_dataset_business.json	yelp_academic_dataset_review.json
1	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "PUFPaY9KxDACqGfsorJp3Q", "review_id": "Ya85v4eqdd6k9Od8HbQjyA", "stars": 4, "date": "2012-08-01", "text": "Mr			
2	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "tU6AxdBYGR4ADwspR98YHA", "review_id": "KpVLNj21_4ubYnctroWdQ0", "stars": 5, "date": "2014-10-13", "text": "Hoagie is an institution. Walking in, it does seem like a throwback to 30 years ago, old			
3	["votes": {"funny": 1, "useful": 1, "cool": 0}, "user_id": "auESFvWw442hbnXgrXAQ0", "review_id": "fF5G6V46Xuwbr37Hnuzl0", "stars": 5, "date": "2015-10-31", "text": "fashioned menu board, booths out of the 70s, and a large selection of food. Their			
4	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "qLczlDzT0_1VB68l0cGvq", "review_id": "pWHt9b_0kxUdFmVrsk2A", "stars": 3, "date": "2015-12-26", "text": "speciality is the Italian Hoagie, and it is voted the best in the area year after year. I			
5	["votes": {"funny": 0, "useful": 1, "cool": 0}, "user_id": "qEESEvV-f-s7YHC0Z4ydJQ", "review_id": "AEy10_Y44isJmNbMtyoMK0", "stars": 2, "date": "2016-04-08", "text": "usually order the burger, while the patties are obviously cooked from frozen, all of the			
6	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "jBoH6qK607wdYy_Yj8QCA", "review_id": "V-bqX62zpxfH2ofKzXPw", "stars": 1, "date": "2016-04-10", "text": "other ingredients are very fresh. Overall, it's a good alternative to Subway, which is down			
7	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "bWrod7Hh_T4qZr-rof0dA", "review_id": "3saJ_LVFUpK4f09r86VNm", "stars": 4, "date": "2016-05-11", "text": "the road.", "type": "review", "business_id": "5UmKMjUEUNdYWqANhGckJw"}]			
8	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "uK8tzradp4MSu3UyrqIBg", "review_id": "D136xALCFmU4dK3M89pg", "stars": 5, "date": "2013-11-08", "text": ".....			
9	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "I_47G-R2_egg7ME5u_ltw", "review_id": "0Lua2_Pbg0Mj09r89-asw", "stars": 3, "date": "2014-03-29", "text": "			
10	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "PP_xoMSYlGr2pb67BbQdA", "review_id": "7N915vbHBH6qguE50AeyA", "stars": 1, "date": "2014-10-29", "text": "			
11	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "jPPPhyFE-UE4532A6K0TVgw", "review_id": "mJC3R33yJUNcL1LCoDU_0", "stars": 4, "date": "2014-11-28", "text": "			
12	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "pL78KCFpk1Dns8a0GepVA", "review_id": "K67wxk8u626yxXUz51060", "stars": 2, "date": "2016-02-24", "text": "			
13	["votes": {"funny": 0, "useful": 1, "cool": 0}, "user_id": "fhNx0MwT1pzj08A9LF8Q", "review_id": "XsA6A0jKwJ0H4FuuAb8X0", "stars": 3, "date": "2012-08-19", "text": "			
14	["votes": {"funny": 0, "useful": 1, "cool": 0}, "user_id": "6rEtoByYjKpUWlXsazx0", "review_id": "rk07Udb09VW3Va6B1-e8H0", "stars": 1, "date": "2013-04-18", "text": "			
15	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "K2uJ1f1And0Wp2Z08Bx0", "review_id": "MeHE-1935L4D1q8s98Wk0", "stars": 1, "date": "2013-07-14", "text": "			
16	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "H9ESvEjGEsRhw0MfKnm0", "review_id": "IS34GjHMKkt9Kc0TJLVEw", "stars": 4, "date": "2013-08-16", "text": "			
17	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "1jWqUJowB69k1arBAu-H7g", "review_id": "5-G0D8Cy7PnqShoBzu8PCA", "stars": 4, "date": "2014-07-11", "text": "			
18	["votes": {"funny": 0, "useful": 5, "cool": 0}, "user_id": "LMBYpcangjBMh4KPz2G0K", "review_id": "Gw6gM231BLGcUcARMBiUf0", "stars": 5, "date": "2012-12-01", "text": "			
19	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "n1Pp3EAegpaAdPxbhR0R", "review_id": "j1Vv_DAsnC0B6meduU4M", "stars": 5, "date": "2013-03-15", "text": "			
20	["votes": {"funny": 0, "useful": 3, "cool": 1}, "user_id": "8fAp1AMHn2M2FJUC0tQ50", "review_id": "3Es8G5jKssusYgeU6_VZp0", "stars": 5, "date": "2013-03-30", "text": "			
21	["votes": {"funny": 0, "useful": 1, "cool": 0}, "user_id": "uK8tzradp4MSu3UyrqIBg", "review_id": "KAKcn7o0PlXK8KsZ-XmktA", "stars": 4, "date": "2013-10-20", "text": "			
22	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "6wV1MSL4_EroGxbnb_92x0", "review_id": "BZJ1KkP8Bxmw02-scqat20", "stars": 5, "date": "2013-11-07", "text": "			
23	["votes": {"funny": 0, "useful": 2, "cool": 1}, "user_id": "345nDw0c-10cplqmwz0", "review_id": "Y0T1D83G5_1PpX0a2RutaA", "stars": 5, "date": "2014-03-22", "text": "			
24	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "u9ULAsnYtGYH65Haj5LMSw", "review_id": "SuyYmiYyIB_wtKtyXDu00", "stars": 4, "date": "2014-09-29", "text": "			
25	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "pdHC0oAcG7gNdhFRAU0Q", "review_id": "zyn_Libz9VZTZ_-0dc4-t0", "stars": 5, "date": "2014-09-29", "text": "			
26	["votes": {"funny": 0, "useful": 3, "cool": 2}, "user_id": "TAKjY3D0XHS1n3D0HfPmQ", "review_id": "uf61rPucUICXhSPALZ1h10", "stars": 5, "date": "2014-11-03", "text": "			
27	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "Kcb_FUG7d_MTzqRL1B3NA", "review_id": "X2g1Y49dnp052R2ZcQc", "stars": 4, "date": "2014-12-18", "text": "			
28	["votes": {"funny": 1, "useful": 1, "cool": 4}, "user_id": "aBULP1N105w-BNvoVUz2w", "review_id": "1W1vq7D10fFxbCA_GWU_A", "stars": 5, "date": "2015-03-06", "text": "			
29	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "ke0045_86ST1Eqd5_AWw", "review_id": "6ASnASw8_XrWdxV29cWGU", "stars": 4, "date": "2015-03-08", "text": "			
30	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "FoF6S21jVUUXptB5_gdN0", "review_id": "RoY1ya0kSY1b5r9w1rrf10", "stars": 4, "date": "2015-03-16", "text": "			
31	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "X85PALUxE0gdrhxAJ2A", "review_id": "ezaz76umnePC8kH6G5f0", "stars": 5, "date": "2015-05-27", "text": "			
32	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "E508w5W1Qd0q0VBmG64kcw", "review_id": "boU6hZwK_AS9Z7YVHV44CA", "stars": 5, "date": "2015-07-02", "text": "			
33	["votes": {"funny": 0, "useful": 1, "cool": 1}, "user_id": "k0a-uli8yr_R1-91pL_RA", "review_id": "e11K8g_0m66pDeR23Tg510", "stars": 5, "date": "2015-11-06", "text": "			
34	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "LQ0y7DSbz2W4tEXNvg2T", "review_id": "MKE48p02_DF19asCpW68A", "stars": 5, "date": "2015-12-04", "text": "			
35	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "r1Kuu--p511fWkxv45Ap", "review_id": "G8Z1XJYp0d0B751H_r4", "stars": 5, "date": "2014-12-15", "text": "			
36	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "q1czlDzT0_1VB68l0cGvq", "review_id": "v59hvg17A2EkFuGkWZx50", "stars": 5, "date": "2015-12-16", "text": "			
37	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "fRMPv1xxMaEAr0u5P2zcw", "review_id": "Pm6hPvRn7dLHpp7Yf9mg", "stars": 3, "date": "2016-01-21", "text": "			
38	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "50umK80_My05t50bMyuNA", "review_id": "vtqE55y9k74q156y0leo0", "stars": 5, "date": "2016-02-22", "text": "			
39	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "891L1VL4_6c9GfjC9Tlu0", "review_id": "q2vQ21a0J_vzL_H8Mkxw", "stars": 5, "date": "2016-03-13", "text": "			
40	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "HAT1Y7g717TP365-hRT080", "review_id": "wJFP1gHK610qaBEypl1RA", "stars": 4, "date": "2016-04-07", "text": "			
41	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "j9wgdb07UtoE7Vb4Mm0w", "review_id": "5FwP610fW_lgpa2n9LzNw", "stars": 5, "date": "2016-05-08", "text": "			
42	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "j9wgdb07UtoE7Vb4Mm0w", "review_id": "Pxr798bJ740zR1MgT_0aw", "stars": 5, "date": "2016-07-08", "text": "			
43	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "X85PALUxE0gdrhxAJ2A", "review_id": "5W1E1FQ0hSR1Nbnh15a", "stars": 5, "date": "2015-05-27", "text": "			
44	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "JRwpl1EnVn3ghy9N0Y1A", "review_id": "XG0bgpRNE0PMY03B9LAthg", "stars": 5, "date": "2016-04-26", "text": "			
45	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "Bk5e309851Y7F5P5aYyg", "review_id": "51dUtnXsCzX0YECjapPha", "stars": 5, "date": "2016-07-10", "text": "			
46	["votes": {"funny": 2, "useful": 2, "cool": 2}, "user_id": "ay9H1Rp1bKaaX0xf17LA", "review_id": "v_uE0bX5P1U0pKPNP4X00", "stars": 4, "date": "2016-10-11", "text": "			
47	["votes": {"funny": 1, "useful": 0, "cool": 1}, "user_id": "jBmXWjLa01kncUk00Majg", "review_id": "18Gux0J_D055f1rNkmcAC", "stars": 4, "date": "2011-02-27", "text": "			
48	["votes": {"funny": 0, "useful": 0, "cool": 1}, "user_id": "4_31U5u1H90m21WbZhnA", "review_id": "2xCd33bp0w6N6hM365v", "stars": 4, "date": "2011-08-15", "text": "			
49	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "bcwr1bfov3P5a1F1Gf9cpg", "review_id": "UrukGXi6mh5Re2fGdxvVPA", "stars": 3, "date": "2011-12-22", "text": "			
50	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "51b5e5-qKdUXfDmC2R00", "review_id": "2_Ru_AS75U038jR0JFF0", "stars": 4, "date": "2013-04-25", "text": "			
51	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "tdaVAr1J5usJL2bq9ydb0", "review_id": "0Z_25SH2bNxxYECZ0FA", "stars": 4, "date": "2013-04-25", "text": "			
52	["votes": {"funny": 0, "useful": 0, "cool": 0}, "user_id": "W_HfP6d0M28rWDRNvY5n", "review_id": "PIH1XT1WBRy9Fnb3hfv010", "stars": 3, "date": "2013-09-17", "text": "			

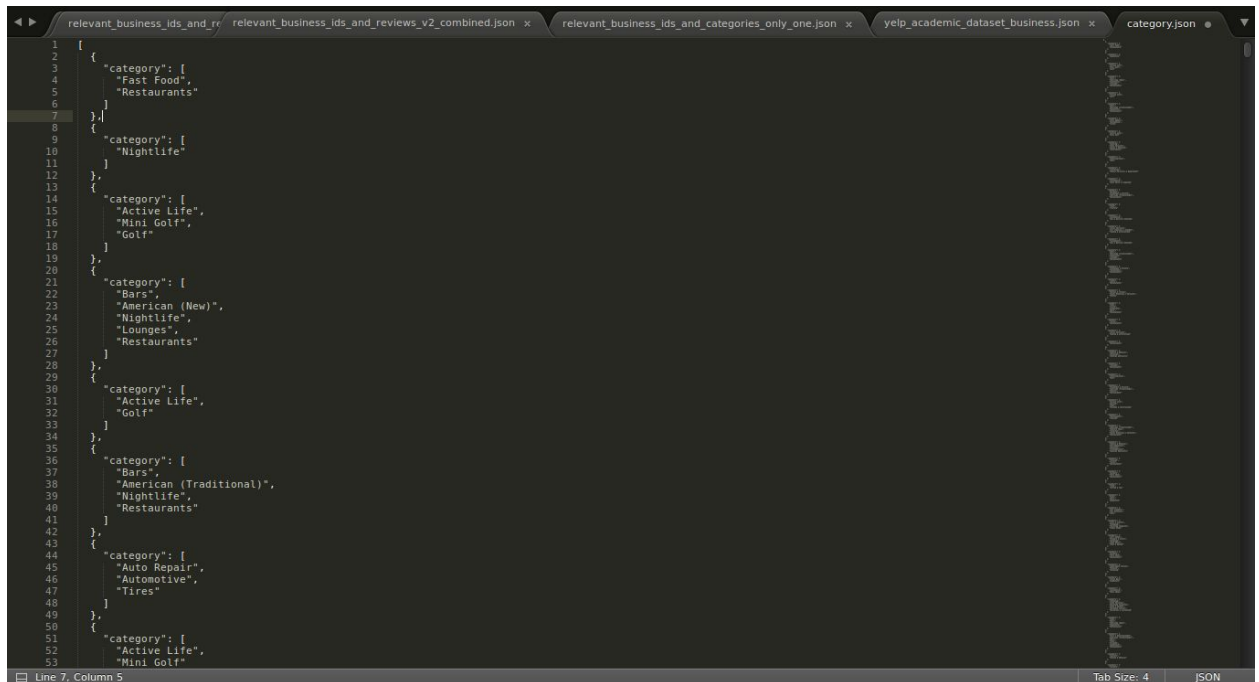
b. Format:

```
{
  "votes": {
    "funny": 0,
    "useful": 0,
    "cool": 0
  },
  "user_id": "PUFPaY9KxDACqGfsorJp3Q",
  "review_id": "Ya85v4eqdd6k9Od8HbQjyA",
  "stars": 4,
  "date": "2012-08-01",
  "text": "Mr Hoagie is an institution. Walking in, it does seem like a throwback to 30 years ago, old fashioned menu board, booths out of the 70s, and a large selection of food. Their speciality is the Italian Hoagie, and it is voted the best in the area year after year. I usually order the burger, while the patties are obviously cooked from frozen, all of the other ingredients are very fresh. Overall, it's a good alternative to Subway, which is down the road.",
  "type": "review",
  "business_id": "5UmKMjUEUNdYWqANhGckJw"
}
```

.....

Here is the sequence of scripts to run to get the finished files:

1. First, we need only business_id and relevant categories from the file A. The category would be our class label. For this, we need to extract all the categories for this dataset and check what all categories we are relevant and their counts.
2. preprocess/category_all.py
 - o Input file: **yelp_academic_dataset_business.json**
 - o Output file: **category.json**



The screenshot shows a code editor with four tabs: 'relevant_business_ids_and_re', 'relevant_business_ids_and_reviews_v2_combined.json', 'relevant_business_ids_and_categories_only_one.json', and 'yelp_academic_dataset_business.json'. The active tab is 'relevant_business_ids_and_categories_only_one.json'. The code is JSON data with line numbers 1 through 53 on the left. The JSON structure consists of an array of objects, each containing a 'category' array. The categories listed are: ['Fast Food', 'Restaurants'], ['Nightlife'], ['Active Life', 'Mini Golf', 'Golf'], ['Bars', 'American (New)', 'Nightlife', 'Lounges', 'Restaurants'], and ['Auto Repair', 'Automotive', 'Tires']. The status bar at the bottom indicates 'Line 7, Column 5', 'Tab Size: 4', and 'JSON'.

```
1 {
2   "category": [
3     "Fast Food",
4     "Restaurants"
5   ]
6 }
7
8 {
9   "category": [
10    "Nightlife"
11  ]
12 }
13
14 {
15   "category": [
16     "Active Life",
17     "Mini Golf",
18     "Golf"
19   ]
20 }
21
22 {
23   "category": [
24     "Bars",
25     "American (New)",
26     "Nightlife",
27     "Lounges",
28     "Restaurants"
29   ]
30 }
31
32 {
33   "category": [
34     "Active Life",
35     "Golf"
36   ]
37 }
38
39 {
40   "category": [
41     "Bars",
42     "American (Traditional)",
43     "Nightlife",
44     "Restaurants"
45   ]
46 }
47
48 {
49   "category": [
50     "Auto Repair",
51     "Automotive",
52     "Tires"
53   ]
54 }
```

- Format:

```
[
  {
    "category": [
      "Fast Food",
      "Restaurants"
    ]
  },
  {
    "category": [
      "Nightlife"
    ]
  },
  {
    "category": [
      "Active Life",
      "Mini Golf",
      "Golf"
    ]
  },
  {
    "category": [
      "Bars",
      "American (New)",
```



```
"Nightlife",  
"Lounges",  
"Restaurants"
```

```
]
```

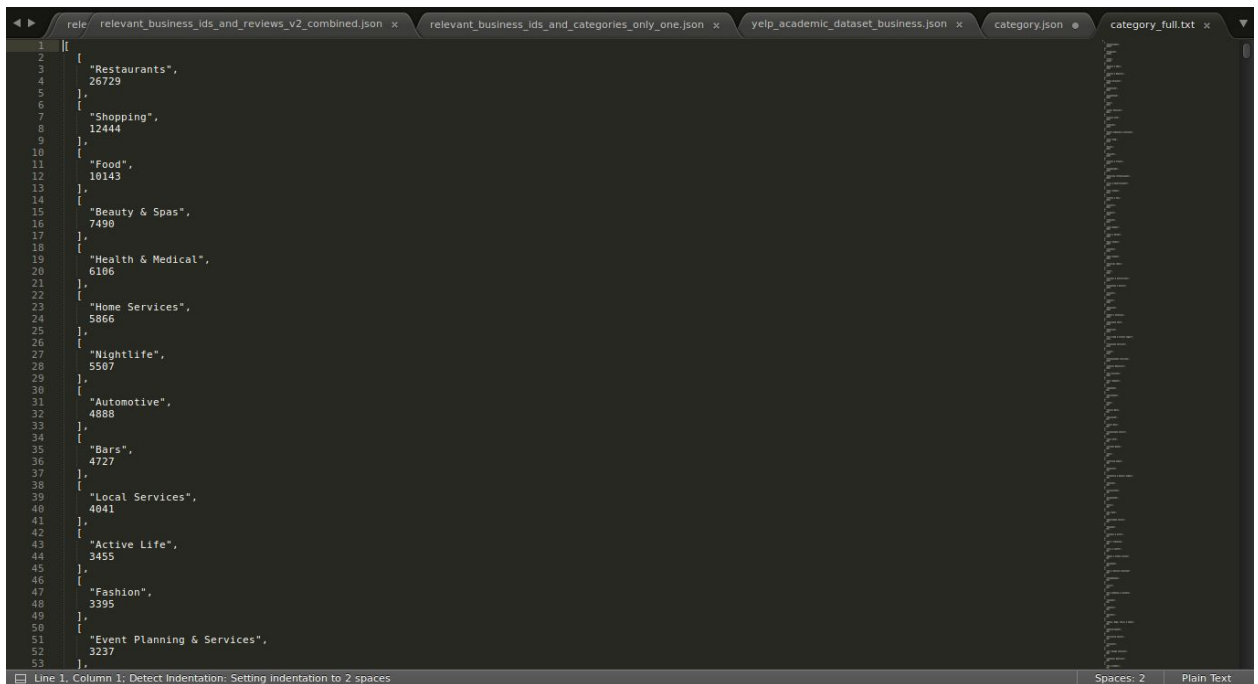
```
},
```

.....

- As we see, many of these are not restaurants. So we need a list of all the unique categories

3. preprocess/class_label_all.py

- Input: **category.json**
- Output: **category_full.txt**



```
1 [{"category": "Restaurants",  
2   "count": 26729},  
3 {"category": "Shopping",  
4   "count": 12444},  
5 {"category": "Food",  
6   "count": 10143},  
7 {"category": "Beauty & Spas",  
8   "count": 7498},  
9 {"category": "Health & Medical",  
10  "count": 6106},  
11 {"category": "Home Services",  
12  "count": 5866},  
13 {"category": "Nightlife",  
14  "count": 5507},  
15 {"category": "Automotive",  
16  "count": 4888},  
17 {"category": "Bars",  
18  "count": 4727},  
19 {"category": "Local Services",  
20  "count": 4041},  
21 {"category": "Active Life",  
22  "count": 3455},  
23 {"category": "Fashion",  
24  "count": 3395},  
25 {"category": "Event Planning & Services",  
26  "count": 3237},  
27 {"category": "Other",  
28  "count": 1000000}],
```

- Format:

```
[  
  [  
    "Restaurants",  
    26729  
  ],  
  [  
    "Shopping",  
    12444  
  ],  
  [  
    "Food",  
    10143
```

```

],
[
  "Beauty & Spas",
  7490
],
[
  "Health & Medical",
  6106
],
]

```

.....

- Since, there were around only 1000 different categories, we had to look at each of them and decide which category belongs to a restaurant or a place that serves food.
- So, we manually went through the list and prepared a list of categories that are relevant to our requirement.
- File: [relevant_labels_raw.txt](#)

```

1 Mexican
2 American (Traditional)
3 Italian
4 Chinese
5 American (New)
6 Japanese
7 Seafood
8 Mediterranean
9 Asian Fusion
10 Thai
11 French
12 Indian
13 Greek
14 Vietnamese
15 Middle Eastern
16 Korean
17 Canadian (New)
18 Latin American
19 Ethnic Food
20 German
21 British
22 Hawaiian
23 Caribbean
24 Spanish
25 Filipino
26 Turkish
27 Portuguese
28 Irish
29 Peruvian
30 Lebanese
31 African
32 Taiwanese
33 Scottish
34 Persian/Iranian
35 Brazilian
36 Kebab
37 Cuban
38 Ethiopian
39 Pretzels
40 Mongolian
41 Cantonese
42 Polish
43 Moroccan
44 Afghan
45 Belgian
46 Russian
47 Colombian
48 Falafel
49 Indonesian
50 Malaysian
51 Arabian
52 New Mexican Cuisine
53 Cambodian

```

- Format:

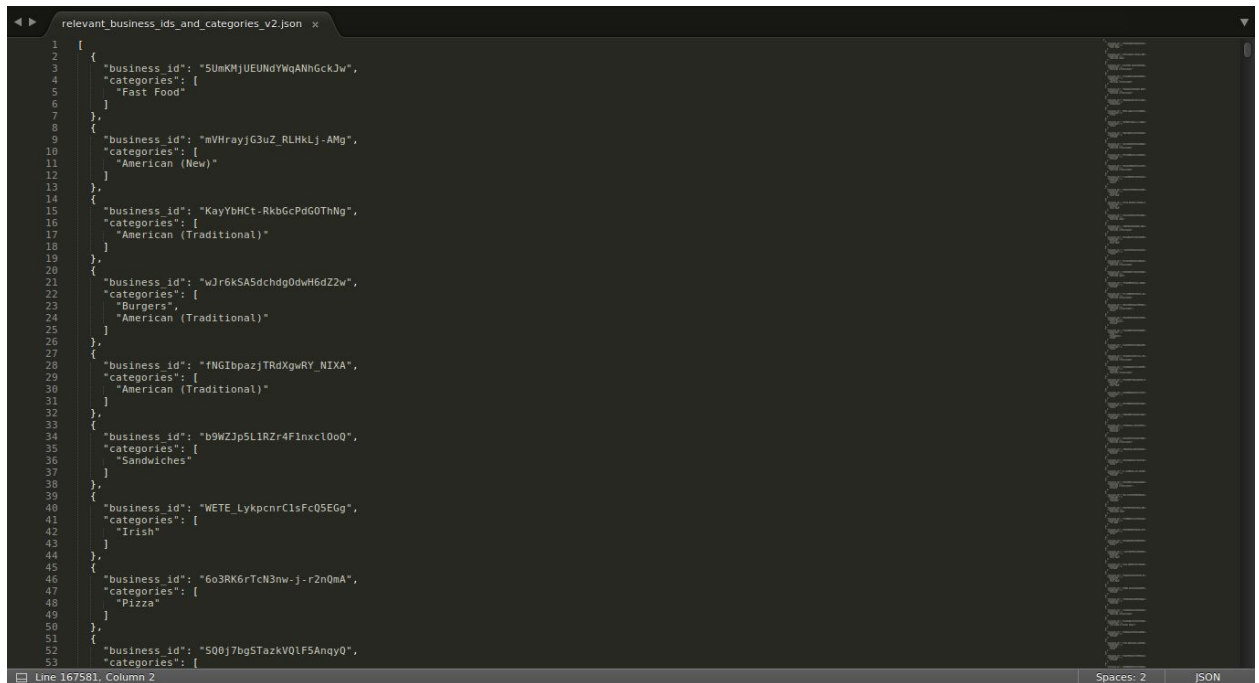
```

Mexican
American (Traditional)
Italian
Chinese
American (New)
Japanese
Seafood
Mediterranean
Asian Fusion

```

Thai
French
Indian
.....

- These all labels are related to restaurants
- 4. Next, we will only keep the data that is related to restaurants and only the reviews related to those restaurants
 - labels/relevant_business_ids_and_categories.py
 - Input: [yelp_academic_dataset_business.json](#)
 - Output: [relevant_business_ids_and_categories.json](#)
 - Format:



```
1  [
2  {
3    "business_id": "5UmKHjUEUNdYwqANhGckJw",
4    "categories": [
5      "Fast Food"
6    ]
7  },
8  {
9    "business_id": "mVHrayjG3uZ_RLHKLj-AMg",
10   "categories": [
11     "American (New)"
12   ]
13 },
14 {
15   "business_id": "KayYbHct-RkbGcPdG0ThNg",
16   "categories": [
17     "American (Traditional)"
18   ]
19 },
20 {
21   "business_id": "wJr6kSA5dchdg0dWH6dZ2w",
22   "categories": [
23     "Burgers",
24     "American (Traditional)"
25   ]
26 },
27 {
28   "business_id": "fNGIbpazjTRdXgwRY_NIXA",
29   "categories": [
30     "American (Traditional)"
31   ]
32 },
33 {
34   "business_id": "b9WZjp5LIaZr4FInxcl0o0",
35   "categories": [
36     "Sandwiches"
37   ]
38 },
39 {
40   "business_id": "WETE_LykpcnrC1sFc05EGg",
41   "categories": [
42     "Irish"
43   ]
44 },
45 {
46   "business_id": "6o3RK6rTcN3nw-j-r2nQmA",
47   "categories": [
48     "Pizza"
49   ]
50 },
51 {
52   "business_id": "5Q0j7bg5Tazkv0lF5Anqy0",
53   "categories": [
```

- 5. Similarly, we only keep reviews of those business units that have relevant categories
 - First, we extract the list of business_ids that we are interested in.
 - labels/relevant_business_ids.py
 - Input: [relevant_labels_raw.txt](#), [yelp_academic_dataset_business.json](#)
 - Output: [relevant_business_ids.txt](#)
 - Format:


```

relevant_business_ids.txt
1 5UmKMjUEUNdYWqANhGckJw
2 mVHrayjG3uZ_RLHklj-AMq
3 KayYbHct-RkBgCpG0TnMq
4 wJr6kASdchq0bH6edZw
5 fNGI0pazjTRdXgwAY_NIXA
6 b9WZjP5LIRZr4FIxc10oQ
7 WETE_LyKpcnrlC1sFc0SEg
8 6a3R6rTcK3w-1-r20m4
9 S00j7bG5TazKVQ1F5AnqyQ
10 wqu7IL0mI0PSduRw0p4AQ
11 P1f32W01mXauZj0Uf4w
12 801nAkJSE6qpfKTVOxrPvQ
13 t_gan0EXAw8csKieFyazJw
14 PK6a51zcKHFwk810xt50A
15 611J8_05xRpeK_8qUp36-g
16 1qCu0cKs5HRv670HovAVpg
17 sbW8qHJgzEI42B05-3New
18 W7ysp8GLC8rs0GKIvWAcBQ
19 -da5f8sf5UKy91CXnySA
20 Mc1kHxxEqZ2X0joaRNKlAw
21 EaAY135VeJr1BzId8lUQwQ
22 TrvWSIAMFZ9zi3y2_K6w0A
23 tu-Ye0839P3m0G5ix_AQ
24 6p9TLP21854agxYMFSDjg
25 JDDeaFb0JXD1NbznSIC9g
26 FlXanVngM79U0VqFH0Z5EA
27 uLnDfng11EroQKLnMnQ
28 eznh0XjVtv924K-KYx_1l0
29 us5R0UMtAgPcxV3Yn00xMQ
30 CL3tZqbY7B5ZgewKCS6-0
31 uZ200A01k0wjoTlYftw
32 53Yk2EODwnV9qHQECzeHVQ
33 iBZRphMDfC5ZaMbawc10vA
34 hnhHt6LinzC-1X6y3GSP7jw
35 CjAVtAtnHuab0blyr8mQg
36 rBBwdbLCW-ZUE537BL0khw
37 m1iHJkpK8nZW_45LlhrbA
38 U--dvIH2j2q-ulb_xKn10g
39 02tp6810cc1Eq080zG0yKw
40 8Nm_jcCYtMYW00SDH1XA
41 9tAbT0dYASHL7EU1e-0Ww
42 f3jBR6nkc7vW6VqxoxQ
43 T49ZV8a6m4X5Hsjke-Ar4
44 e0v0Bq2E8s0CKMaqTImw
45 js3FamFVPR1o9eEVExbA
46 Lnl0-yTh0Hr5P91r5Kw0A
47 JX2g0F7Zuy2UcuKpPcKT-1A
48 QVZBd_eu0oJ4IN0m02XPTQ
49 7uTS3z20z5Hv9BqtBapqw
50 ucK010v0c3xETLhNpKc5w
51 Rjx8GcWuCzfvskC51P1c0A
52 ZL6DNzenFmKpz6uADLNg
53 0cTG-KzDleuRub-pb6J530

```

- Now, we use these business_id to keep only the reviews related to these business_id
- labels/relevant_business_ids_and_reviews.py
 - Input: **relevant_business_ids.txt, yelp_academic_dataset_review.json**
 - Output: **relevant_business_ids_and_reviews.json**
 - Format:

```
[
{
  "business_id": "5UmKMjUEUNdYWqANhGckJw",
  "review": "Mr Hoagie is an institution. Walking in, it does seem like a throwback to 30 years ago, old fashioned menu board, booths out of the 70s, and a large selection of food. Their speciality is the Italian Hoagie, and it is voted the best in the area year after year. I usually order the burger, while the patties are obviously cooked from frozen, all of the other ingredients are very fresh. Overall, its a good alternative to Subway, which is down the road."
},

```

- Now, some of the restaurants have more than one relevant category, but since this is a class label, we must restrict each restaurant to just one category. For this, we will see which category of the restaurant is more popular among the reviews.
 - First, we combine all the reviews of a business_id under one business_id. During this process, we remove any numbers and symbols and convert each word to lowercase, so that comparison is uniform.
 - labels/relevant_business_ids_and_reviews_combined.py
 - Input: **relevant_business_ids_and_reviews.json**

■ Output: [relevant_business_ids_and_reviews_combined.json](#)

```
1 {
2   "sgBI3UDEcNYKwuUb92CYdA": [
3     "this",
4     "is",
5     "the",
6     "best",
7     "dim",
8     "sum",
9     "in",
10    "entire",
11    "arizona",
12    "the",
13    "owner",
14    "was",
15    "the",
16    "cookers",
17    "of",
18    "great",
19    "wall",
20    "on",
21    "th",
22    "ave",
23    "phoenix",
24    "they",
25    "moved",
26    "to",
27    "chandler",
28    "and",
29    "start",
30    "their",
31    "own",
32    "restaurant",
33    "if",
34    "you",
35    "want",
36    "the",
37    "best",
38    "chinese",
39    "traditional",
40    "dim",
41    "sum",
42    "you",
43    "cant",
44    "miss",
45    "this",
46    "one",
47    "we",
48    "used",
49    "to",
50    "drive",
51    "mile",
52    "onnewav".
53  ]
54 }
```

■ Format:

```
{
  "sgBI3UDEcNYKwuUb92CYdA": [
    "this",
    "is",
    "the",
    "best",
    "dim",
    "sum",
    "in",
    "entire",
    "arizona",
    "the",
    "owner",
    "was",
    "the",
    "cookers",
    "of",
    "great",
    "wall",
    "on",
    "th",
    "ave",
```

"phoenix",
 "they",
 "moved",
 "to",
 "chandler",
 "and",
 "start",
 "their",
 "own",
 "restaurant",
 "if",
 "you",
 "want",
 "the",
 "best",

.....

7. Now, we will process the file with relevant business_id to keep only one label per business_id.

- labels/only_one_category_business_ids.py
 - Input: [relevant_business_ids_and_reviews_combined.json](#)
 - Output: [relevant_business_ids_and_categories_only_one.json](#)

```

1  [
2  {
3    "category": "Fast Food",
4    "business_id": "5UmKJUEUdYwqAhhGckJw"
5  },
6  {
7    "category": "American (New)",
8    "business_id": "mVhrayjG3uz_RLHkLj-AMg"
9  },
10 {
11   "category": "American (Traditional)",
12   "business_id": "KayYbHct-RkbGcPoG0Thkg"
13 },
14 {
15   "category": "American (Traditional)",
16   "business_id": "wJr6kSA5dchdg0dwH6dZ2w"
17 },
18 {
19   "category": "American (Traditional)",
20   "business_id": "rNGIbpozjTrdXgwRY_NIXA"
21 },
22 {
23   "category": "Sandwiches",
24   "business_id": "b9wZJp5LiRzr4Flnxc10o0"
25 },
26 {
27   "category": "Irish",
28   "business_id": "WETE_LykcpcnrC1sFcQ5EGg"
29 },
30 {
31   "category": "Pizza",
32   "business_id": "6o3RK6rTcn3nw-j-r2nQmA"
33 },
34 {
35   "category": "Chinese",
36   "business_id": "500j7bg5TazkV0lF5Anqy0"
37 },
38 {
39   "category": "American (Traditional)",
40   "business_id": "wqu7lLomI0PsdurW0Wp4A0"
41 },
42 {
43   "category": "Italian",
44   "business_id": "P1fJb2W0ImXoIudj8UE44w"
45 },
46 {
47   "category": "Pizza",
48   "business_id": "t_gan0EXAw8csKieFyazJw"
49 },
50 {
51   "category": "Fast Food",
52   "business_id": "PK6a51zckHFWk8l0oxt5DA"
53 },
54 ]
  
```

- Format:

```
[
{
```

```

"category": "Fast Food",
"business_id": "5UmKMjUEUNdYWqANhGckJw"
},
{
"category": "American (New)",
"business_id": "mVHrayjG3uZ_RLHkLj-AMg"
},
{
"category": "American (Traditional)",
"business_id": "KayYbHCt-RkbGcPdGOTnNg"
},

```

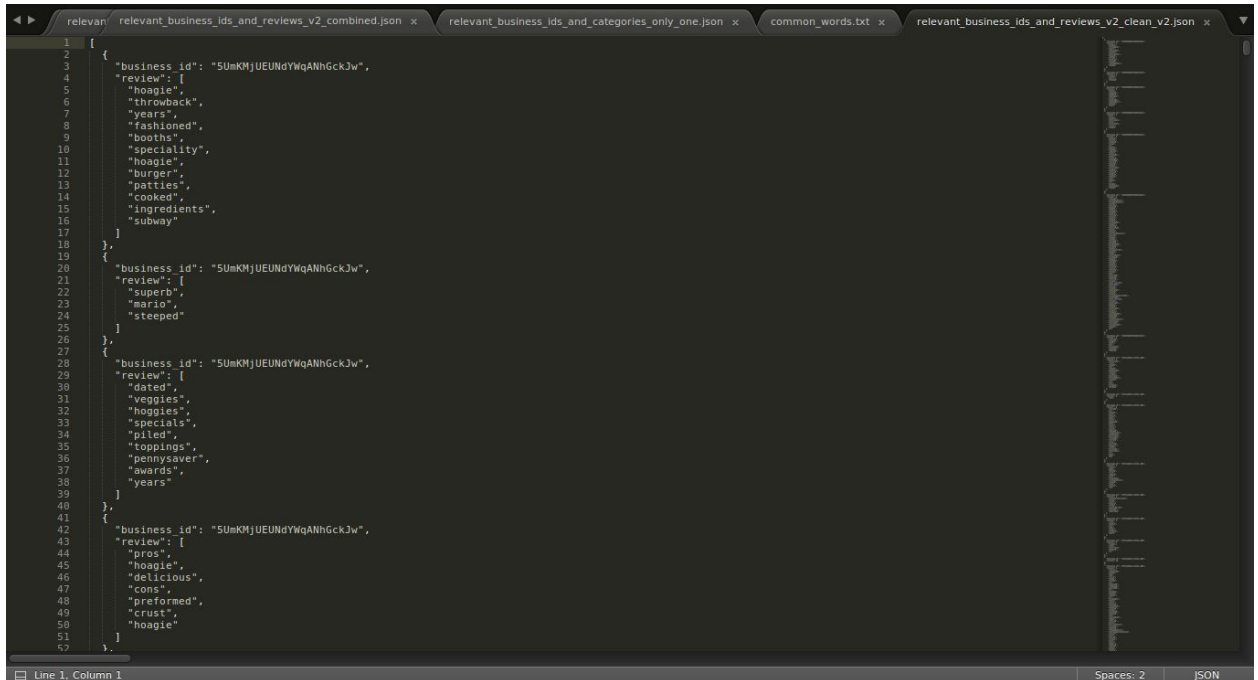
8. Now, we clean our reviews file and keep it ready to be used further in the process.
 - For this, there would be two approaches. The first approach removes all the common words, stop words, numbers, symbols and converts all reviews into lowercase words, so that comparison is uniform.
 - We have gathered a list of common words combined from 2 sources:
 - 5000 words free list from <http://www.wordfrequency.info/free.asp>
 - 6000 most frequent english words: <http://www.insightin.com/esl/>
 - Our list contains a total of 6994 unique words
 - common_words.txt

```

1 brew
2 transit
3 allies
4 landing
5 extraordinary
6 aluminum
7 gesture
8 unity
9 cue
10 acknowledge
11 effort
12 drag
13 attitude
14 community
15 relax
16 maintaining
17 el
18 nutrient
19 latter
20 rehabilitation
21 held
22 fair
23 laughing
24 manager
25 stimulate
26 castle
27 wally
28 increased
29 drive
30 diversity
31 writer
32 admit
33 kennedy
34 rub
35 veil
36 manual
37 skip
38 sociology
39 moonlight
40 enterprise
41 heard
42 hat
43 mighty
44 conventional
45 negotiation
46 rude
47 substrate
48 portion
49 appointment
50 predator
51 shouldn't
52 tangent
53 hockey

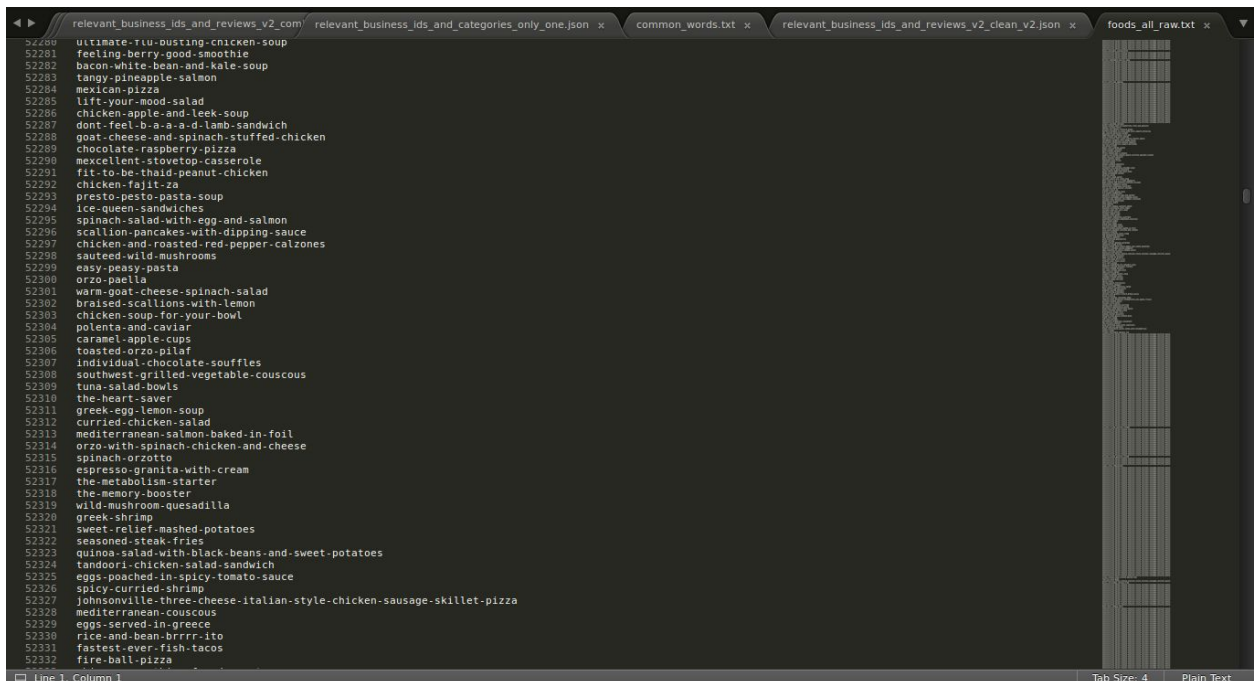
```

- cleansing/relevant_bids_reviews_clean.py
 - Input: [relevant_business_ids_and_reviews.json](#)
 - Output: [relevant_business_ids_and_reviews_clean.json](#)



The screenshot shows a code editor with several tabs. The active tab is 'relevant_business_ids_and_reviews_v2_combined.json'. It displays a JSON array of business reviews. Each review object contains a 'business_id' and a 'review' array of words. The words are food-related terms like 'hoagie', 'throwback', 'years', 'fashioned', 'booths', 'specialty', 'burger', 'patties', 'cooked', 'ingredients', 'subway', 'dated', 'vegies', 'hoggies', 'specials', 'piled', 'toppings', 'pennysaver', 'awards', 'years', 'pros', 'delicious', 'cons', 'preformed', 'crust', and 'hoagie'.

- For the second approach, we only keep food related words in our reviews. To gather food related words, we scrapped the website <http://allrecipes.com/> for all the world's recipies.
- scrap/food_scrapper.py
 - The script scrap the website and pings about 300,000 different URLs to get all the recipies.
 - Output: [foods_all_raw.txt](#)



The screenshot shows a code editor with several tabs. The active tab is 'foods_all_raw.txt'. It displays a list of recipes, each preceded by a line number. The recipes include: 'ultimate-ruu-busting-cnicken-soup', 'feeling-very-good-smoothie', 'bacon-white-bean-and-kale-soup', 'tanga-pineapple-salmon', 'mexican-pizza', 'lift-your-mood-salad', 'chicken-apple-and-leek-soup', 'dont-feel-b-a-a-d-lamb-sandwich', 'goat-cheese-and-spinach-stuffed-chicken', 'chocolate-raspberry-pizza', 'mexcellent-stovetop-casserole', 'fit-to-be-thaid-peanut-chicken', 'chicken-fajit-za', 'presto-pesto-pasta-soup', 'ice-queen-sandwiches', 'spinach-salad-with-egg-and-salmon', 'scallion-pancakes-with-dipping-sauce', 'chicken-and-roasted-red-pepper-calzones', 'sauteed-wild-mushrooms', 'easy-peasy-pasta', 'orzo-paella', 'wara-goat-cheese-spinach-salad', 'braised-scallions-with-lemon', 'chicken-soup-for-your-bowl', 'polenta-and-caviar', 'caramel-apple-cups', 'toasted-orzo-pilaf', 'individual-chocolate-souffles', 'southwest-grilled-vegetable-couscous', 'tuna-salad-bowls', 'the-heart-saver', 'greek-egg-lemon-soup', 'curried-chicken-salad', 'mediterranean-salmon-baked-in-foil', 'orzo-with-spinach-chicken-and-cheese', 'spinach-orzotto', 'espresso-granita-with-cream', 'the-metabolism-starter', 'the-memory-booster', 'wild-mushroom-quesadilla', 'greek-shrimp', 'sweet-relief-mashed-potatoes', 'seasoned-steak-fries', 'quinoa-salad-with-black-beans-and-sweet-potatoes', 'tandoori-chicken-salad-sandwich', 'eggs-poached-in-spicy-tomato-sauce', 'spicy-curved-shrimp', 'johnsonville-three-cheese-italian-style-chicken-sausage-skillet-pizza', 'mediterranean-couscous', 'eggs-served-in-greece', 'rice-and-bean-brrrrr-ito', 'fastest-ever-fish-tacos', and 'fire-boll-pizza'.

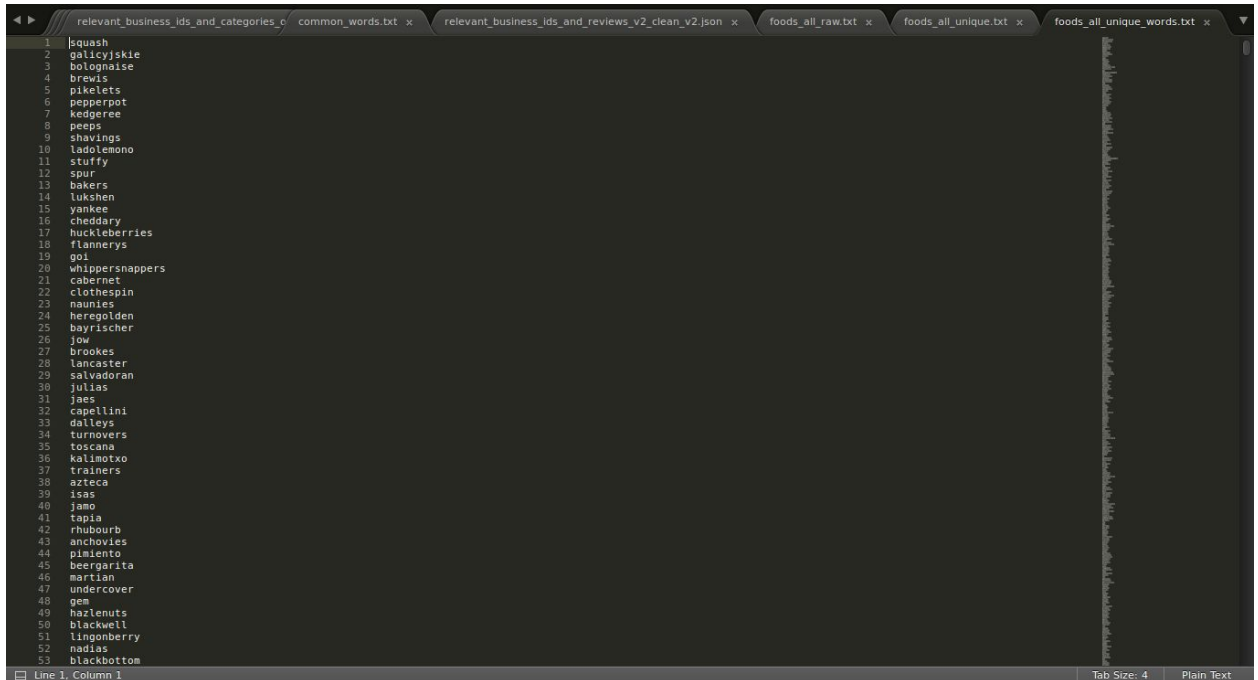
- We gathered about 218,924 recipes.

- Next, we remove the duplicates and keep only unique recipes.
- cleansing/food_words_unique_fulls.py
 - Input: **foods_all_raw.txt**
 - Output: **foods_all_raw_unique.txt**

The screenshot shows a code editor with several tabs open. The active tab is 'foods_all_raw_unique.txt', which contains a list of 53 unique recipes, each on a new line. The recipes are:

- 1 |grilled-chicken-with-cilantro-butter
- 2 |lemon-herb-quinoa
- 3 |california-thai-flank-steak
- 4 |chocolate-pecan-delite
- 5 |sylvias-ribs
- 6 |salmon-roasted-with-tomatoes-and-olives
- 7 |monkey-bread-with-butterscotch-pudding
- 8 |terris-veal-marsala
- 9 |lubeck-marzipan
- 10 |roasted-red-pepper-sauce
- 11 |ramp-potatoes
- 12 |peruvian-chicken-and-rice
- 13 |white-chocolate-holiday-spritz-cookies
- 14 |quick-cranberry-relish
- 15 |mozzarella-and-pesto-crescent-tarts
- 16 |hard-cider-onions-for-hot-dogs-or-sausages
- 17 |pizza-alfredo-with-roasted-vegetables
- 18 |pepper-jelly
- 19 |new-orleans-stew-with-smoked-andouille-chicken-sausage
- 20 |dark-chocolate-cream-cheese-cake
- 21 |fruit-n-yogurt-parfait-with-nutella
- 22 |black-beans-con-jalapeno
- 23 |kabachkovyye-oladi-sladkiye-sweet-zucchini-pancakes
- 24 |white-chocolate-cranberry-cheesecake-cookie-bars
- 25 |lemon-lush
- 26 |chicken-and-pumpkin-pizza
- 27 |creamy-ham-and-bean-soup-gluten-free
- 28 |purple-fiddle-hummus
- 29 |trail-mix-bars
- 30 |spring-vegetable-frittata-for-mother
- 31 |slow-cooker-bananas-foster
- 32 |new-york-state-special-milk-punch
- 33 |prize-cookies
- 34 |lemon-parsley-sole
- 35 |black-bean-lasagna-ii
- 36 |zucchini-bruschetta
- 37 |hearty-hot-chili-dip
- 38 |abbys-super-zucchini-loaf
- 39 |fruitcake-cookies-ii
- 40 |water-chestnut-wraparounds
- 41 |easy-au-gratin-potatoes
- 42 |casa-dressing
- 43 |spaghetti-with-red-clam-sauce
- 44 |easy-pina-colada-french-toast
- 45 |corn-relish
- 46 |cranberry-orange-oatmeal-cookies
- 47 |group-effort-pasta
- 48 |colorful-frosting
- 49 |hoisin-ginger-dressing
- 50 |sugar-free-peanut-butter-balls
- 51 |old-fashioned-cinnamon-nut-cake
- 52 |black-bean-turkey-burgers
- 53 |salmon-casserole

- We got 77,830 unique recipes
- Next step is to extract only food words out of these recipes. We remove common words and stopwords.
- cleansing/food_words_unique_words.py
 - Input: **foods_all_raw_unique.txt**
 - Output: **foods_all_raw_words.txt**

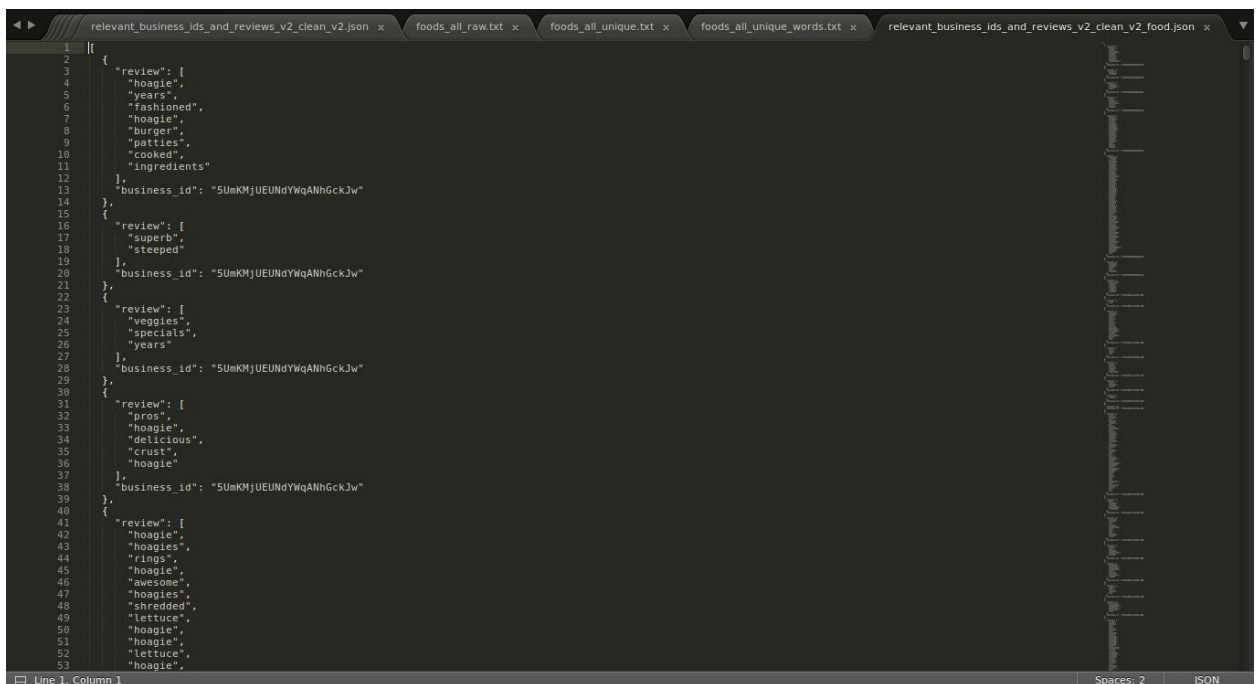


```
1 squash
2 galicyjskie
3 bolognaise
4 brevis
5 pikelets
6 pepperpot
7 kedgeree
8 peeps
9 shavings
10 ladolemono
11 sturffy
12 spur
13 bakers
14 lukshen
15 yankee
16 cheddary
17 huckleberries
18 flannerys
19 got
20 whippersnappers
21 cabernet
22 clothespin
23 nautes
24 heregolden
25 bayrischer
26 jow
27 brookes
28 lancaster
29 salvadoran
30 julias
31 joes
32 capellini
33 dalleys
34 turnovers
35 toscana
36 kalimotxo
37 trainers
38 azteca
39 isas
40 jamo
41 tapia
42 rhubourb
43 anchovies
44 pimiento
45 beergarita
46 marlian
47 undercover
48 gem
49 hazlenuts
50 blackwell
51 lingonberry
52 nadias
53 blackbottom
```

- We get 11,033 unique food words, which covers all the cuisines on allrecipes.com

9. Now, we process our reviews file and keep only food words in the reviews

- cleansing/relevant_bids_reviews_foodclean.py
 - Input: [relevant_business_ids_and_reviews_clean.json](#), [foods_all_raw_words.txt](#)
 - Output: [relevant_business_ids_and_reviews_clean_food.json](#)



```
1 {
2   "review": [
3     "hoagie",
4     "years",
5     "fashioned",
6     "hoagie",
7     "burger",
8     "patties",
9     "cooked",
10    "ingredients"
11  ],
12  "business_id": "5UmKMjUEUNdYWqANhGckJw"
13 },
14 {
15   "review": [
16     "superb",
17     "steeped"
18   ],
19   "business_id": "5UmKMjUEUNdYWqANhGckJw"
20 },
21 {
22   "review": [
23     "veggies",
24     "specials",
25     "years"
26   ],
27   "business_id": "5UmKMjUEUNdYWqANhGckJw"
28 },
29 {
30   "review": [
31     "pros",
32     "hoagie",
33     "delicious",
34     "crust",
35     "hoagie"
36   ],
37   "business_id": "5UmKMjUEUNdYWqANhGckJw"
38 },
39 {
40   "review": [
41     "hoagie",
42     "hoagies",
43     "rings",
44     "hoagie",
45     "awesome",
46     "hoagies",
47     "shredded",
48     "lettuce",
49     "hoagie",
50     "hoagie",
51     "lettuce",
52     "hoagie"
53   ],
54   "business_id": "5UmKMjUEUNdYWqANhGckJw"
55 }
```

10. In the next step is creating the BOW(Bag of words). We generate two different files, one is with each business and the concatenation of all the reviews without filtering any words

for that business and second file is the business id associated with all the review words which are related to food. These files are generated by the script

Name: preprocessing_one_cat.py

Input: relevant_business_ids_and_categories_only_one.json
relevant_business_ids_and_categories_only_one.json

Output: Preprocessed_one_cat_bow.tsv

11. The final step is to build the classifier model and validate it. We have followed the approach of KNN classification with 10-fold validation.

Name: preprocessing_one_cat.py

Input: preprocessed_one_cat_bow.tsv

12. The other approach is running our classification model on data set of reduced dimensionality. So we extracted the random features from both the files generated in step 10.