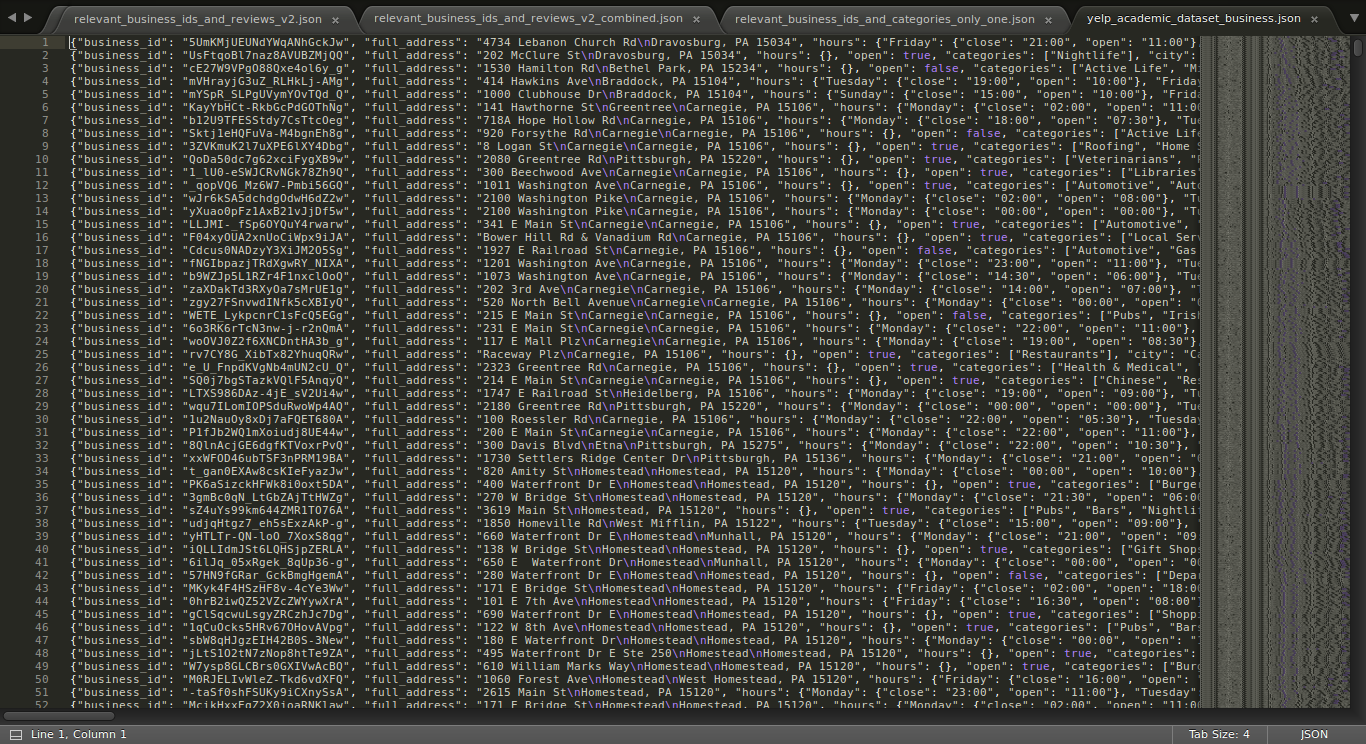
**Readme Document**

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:

1. yelp\_academic\_dataset\_business.json
   1. Contains a list of restaurants, and their categories



* 1. 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,*

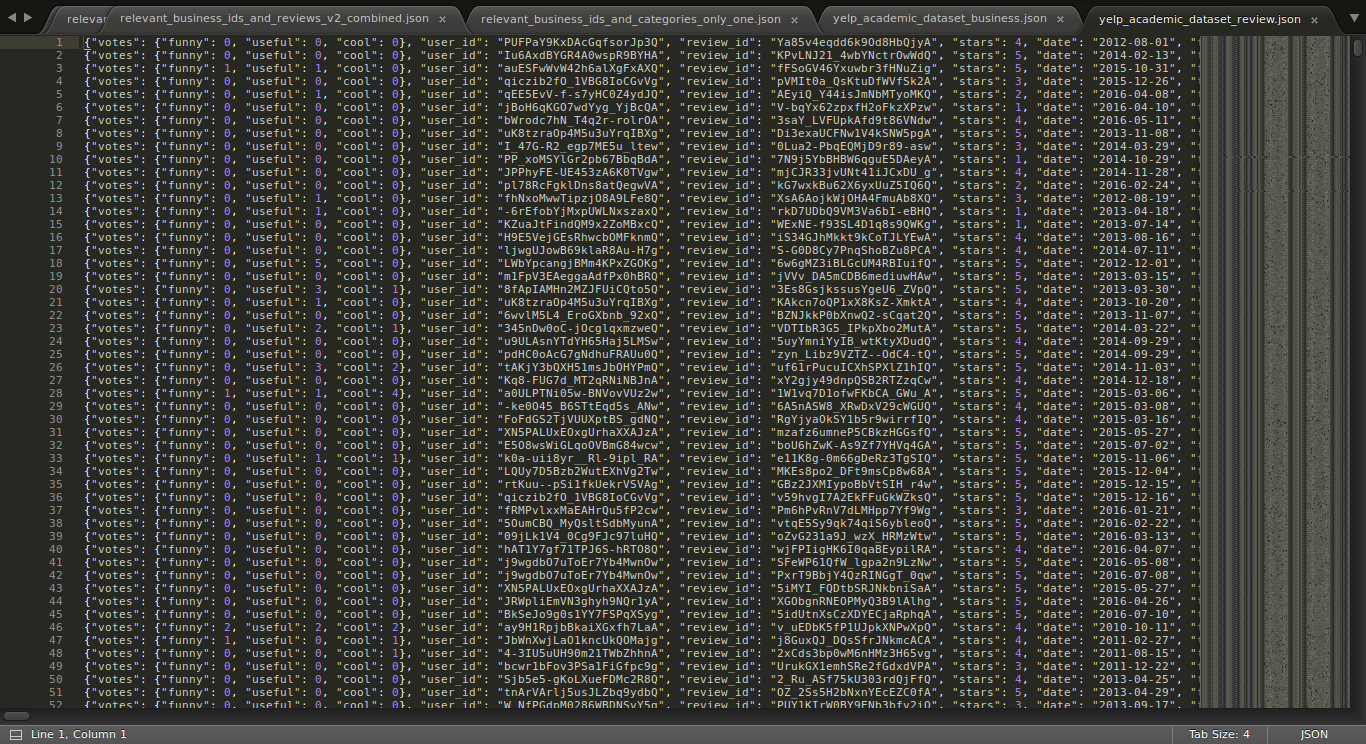
*"Price Range": 1*

*},*

*"type": "business"*

*}*

1. yelp\_academic\_dataset\_review.json
   1. Contains the reviews of the restaurants



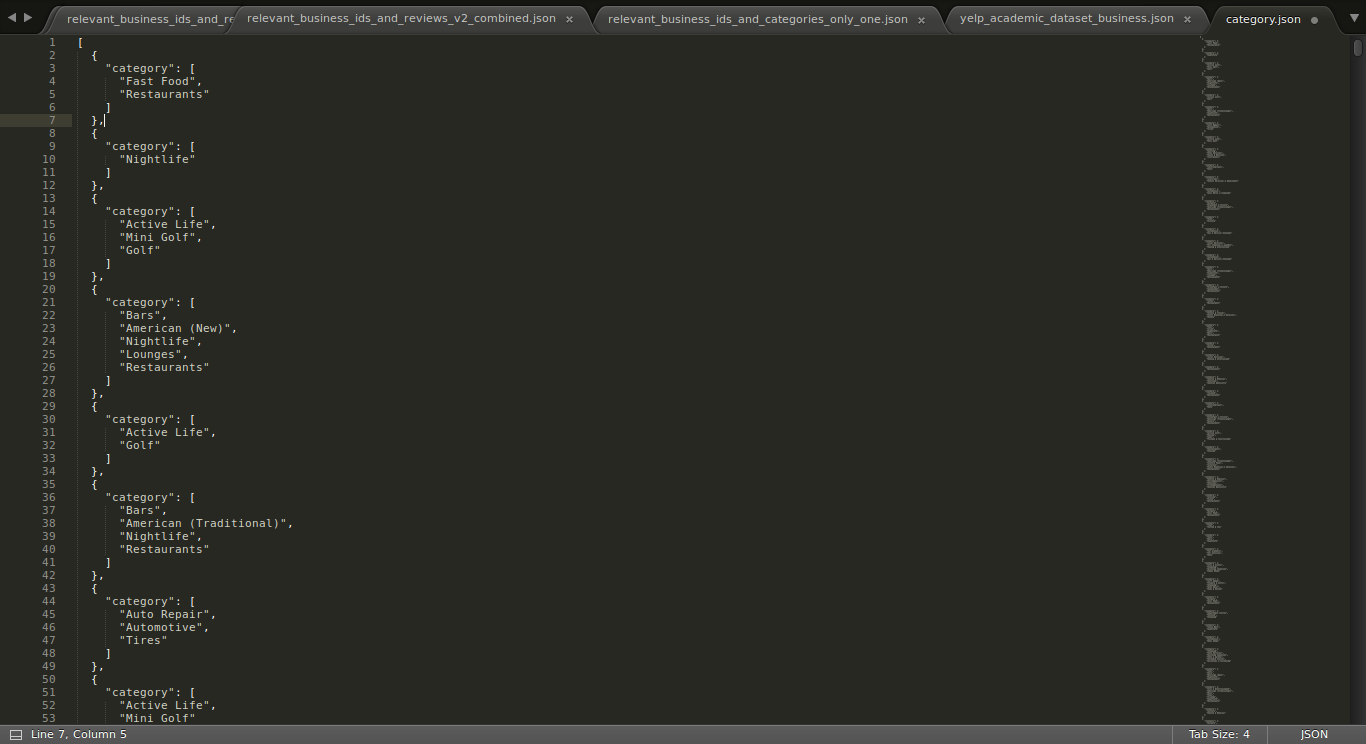
* 1. Format:

{"votes": {"funny": 0, "useful": 0, "cool": 0}, "user\_id": "PUFPaY9KxDAcGqfsorJp3Q", "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, its 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
   * Input file: **yelp\_academic\_dataset\_business.json**
   * Output file: **category.json**



* + Format:

[

{

"category": [

"Fast Food",

"Restaurants"

]

},

{

"category": [

"Nightlife"

]

},

{

"category": [

"Active Life",

"Mini Golf",

"Golf"

]

},

{

"category": [

"Bars",

"American (New)",

"Nightlife",

"Lounges",

"Restaurants"

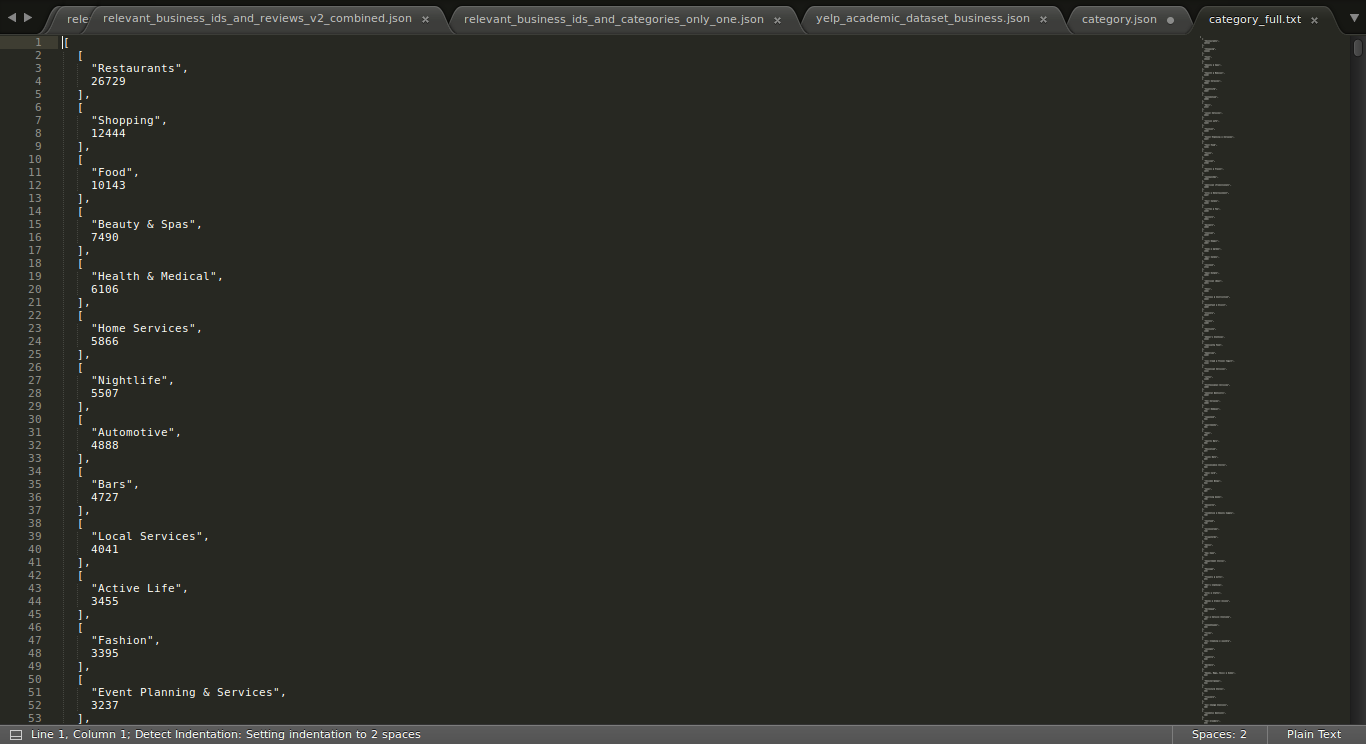
]

},

…...

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

1. preprocess/class\_label\_all.py
   * Input: **category.json**
   * Output: **category\_full.txt**



* + Format:

[

[

"Restaurants",

26729

],

[

"Shopping",

12444

],

[

"Food",

10143

],

[

"Beauty & Spas",

7490

],

[

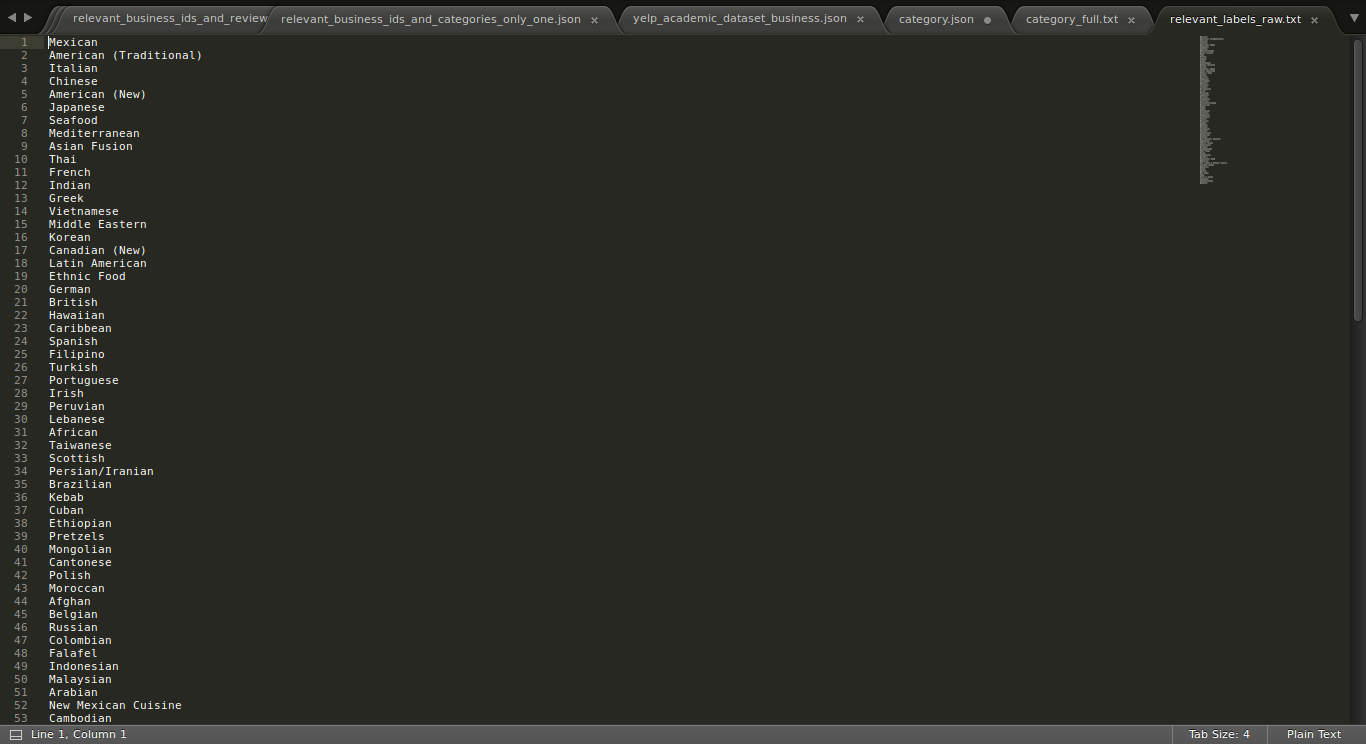
"Health & Medical",

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**



* + Format:

Mexican

American (Traditional)

Italian

Chinese

American (New)

Japanese

Seafood

Mediterranean

Asian Fusion

Thai

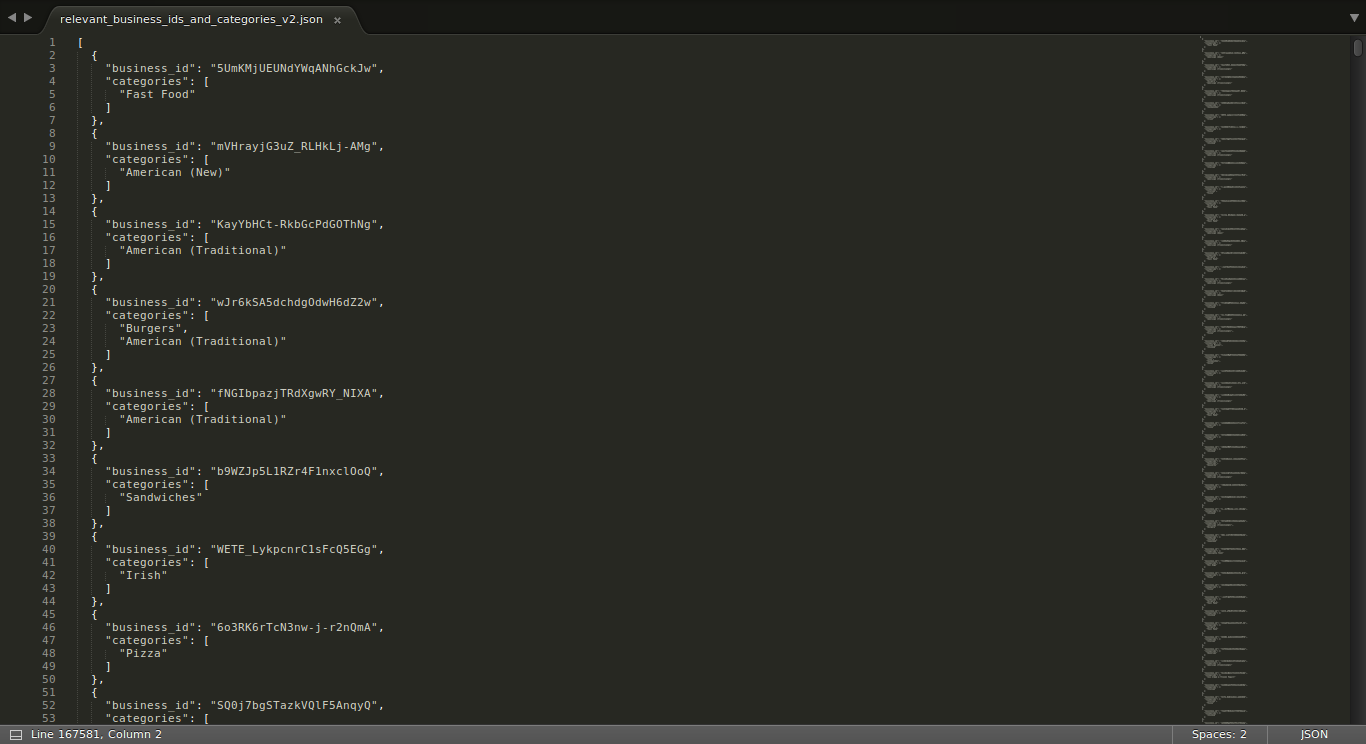
French

Indian

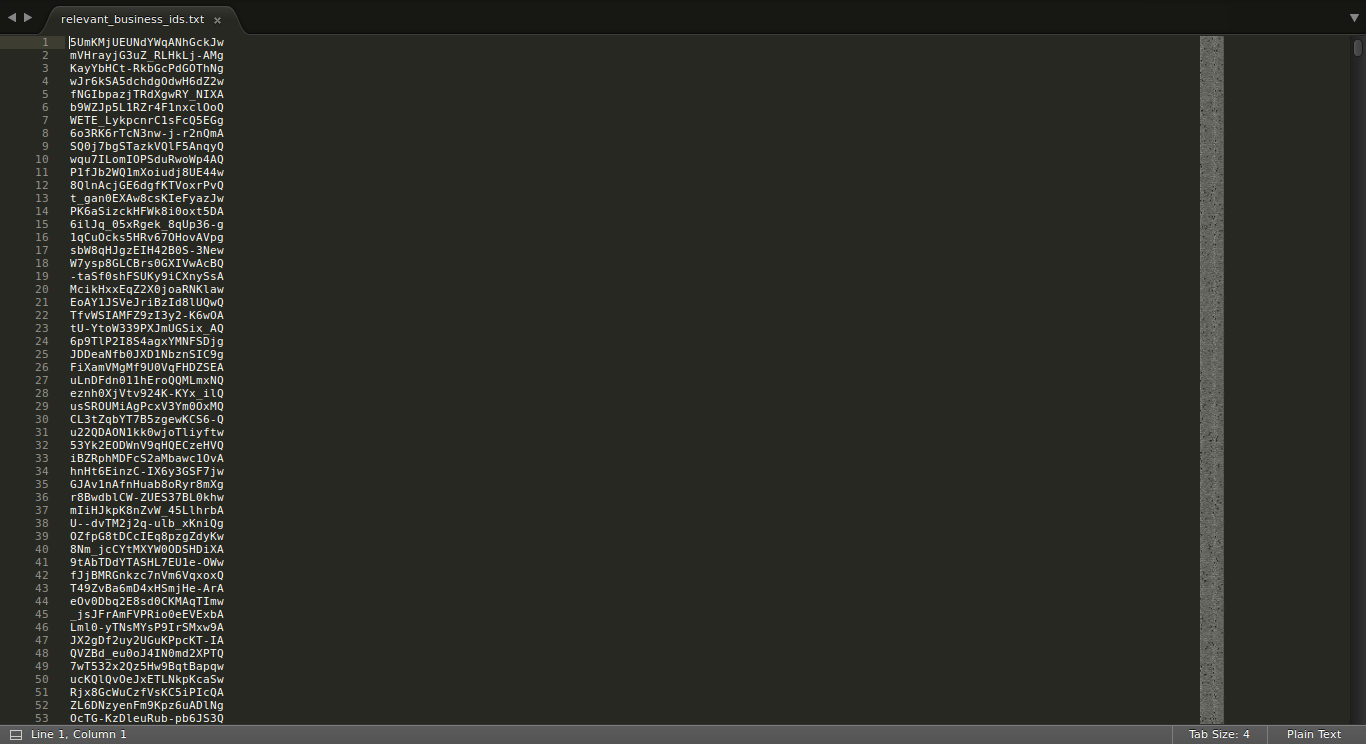
……

* + These all labels are related to restaurants

1. 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. 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:



* + 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:

[

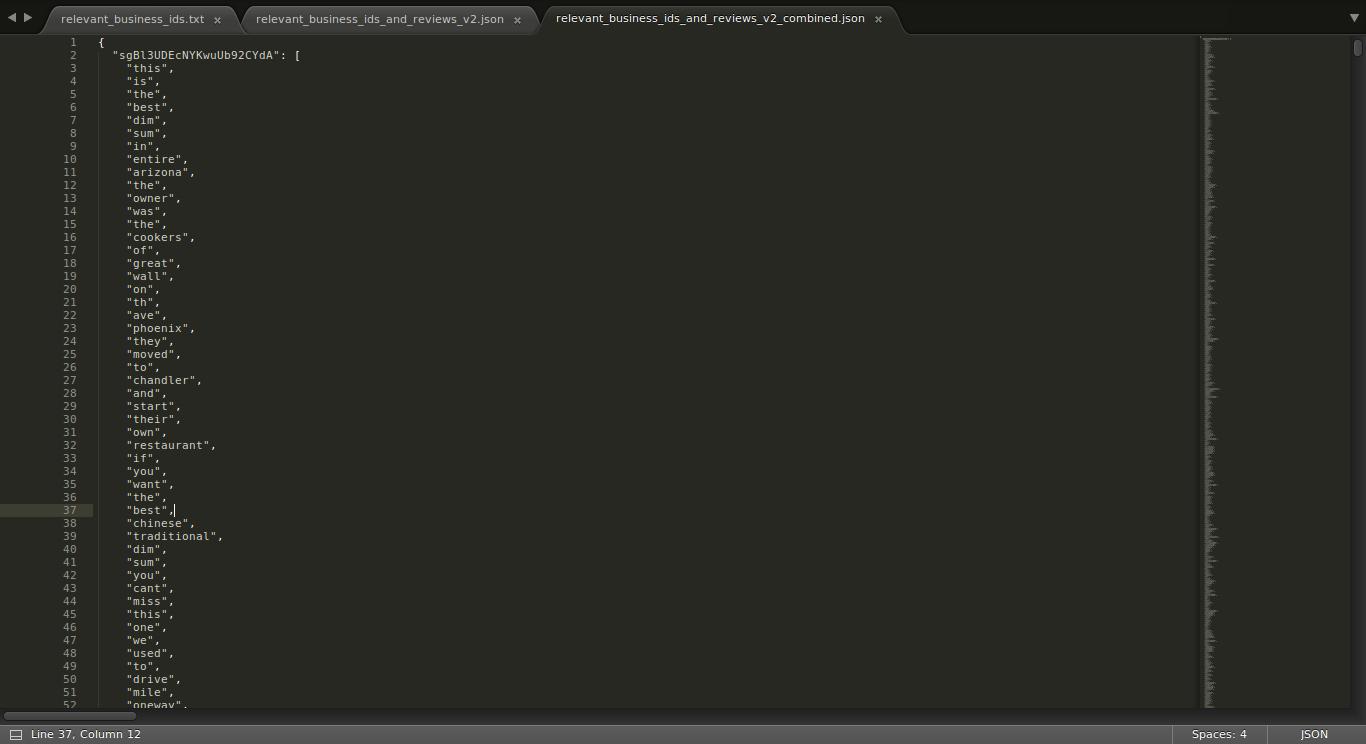
{

"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."

},

1. 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**



* + - Format:

{

"sgBl3UDEcNYKwuUb92CYdA": [

"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",

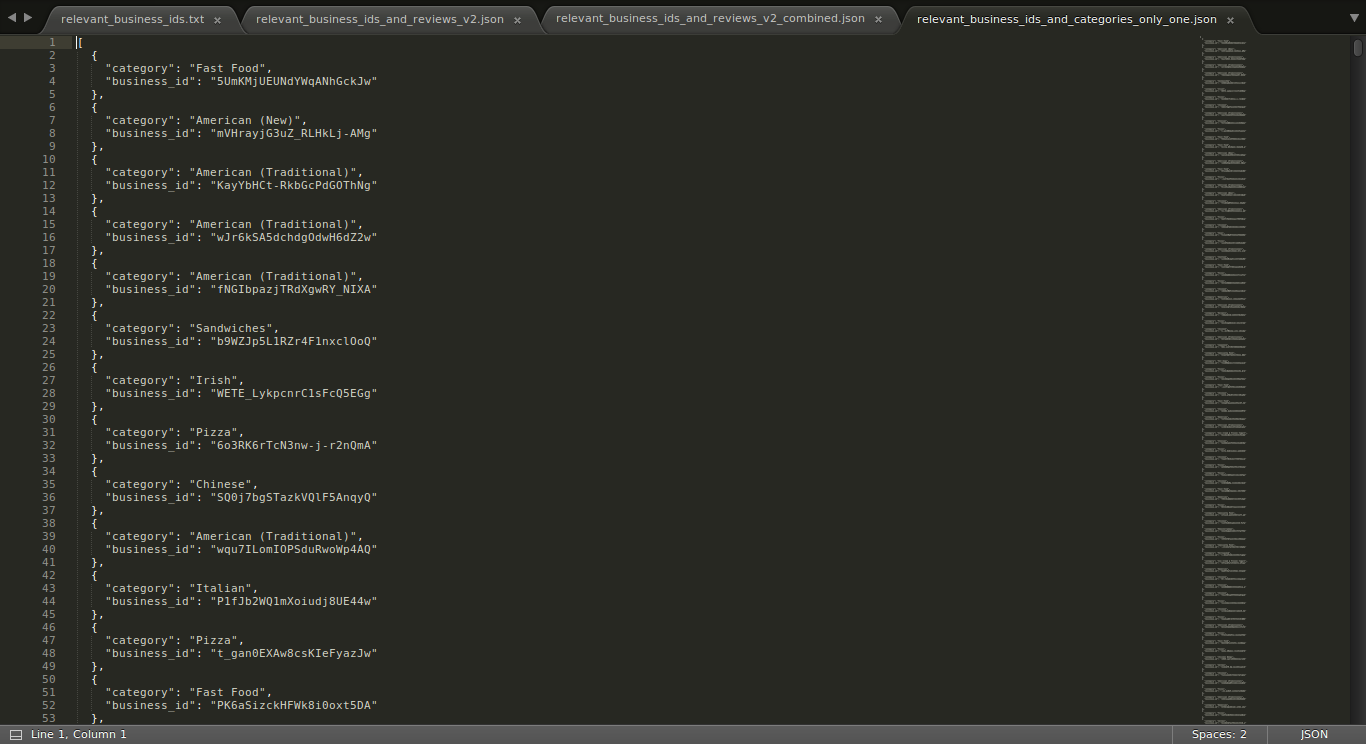
"want",

"the",

"best",

…..

1. 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**



* + - Format:

[

{

"category": "Fast Food",

"business\_id": "5UmKMjUEUNdYWqANhGckJw"

},

{

"category": "American (New)",

"business\_id": "mVHrayjG3uZ\_RLHkLj-AMg"

},

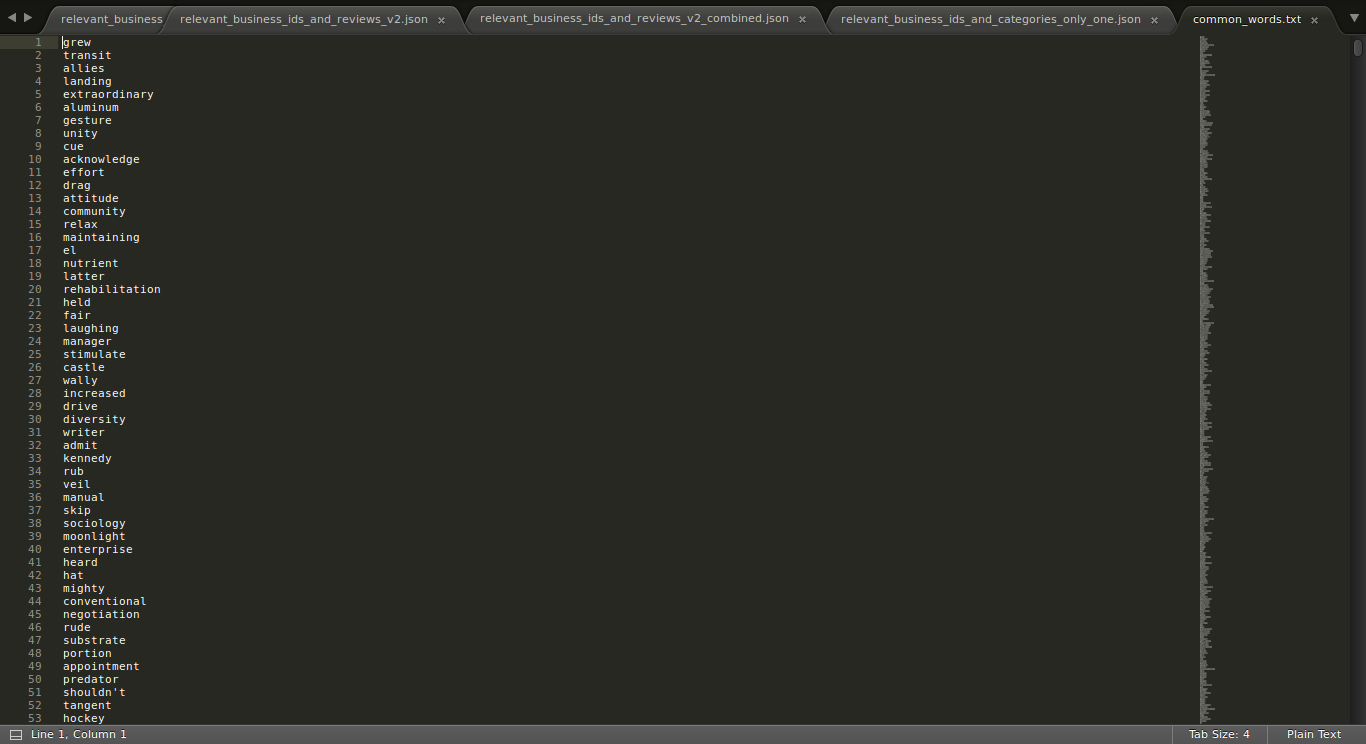
{

"category": "American (Traditional)",

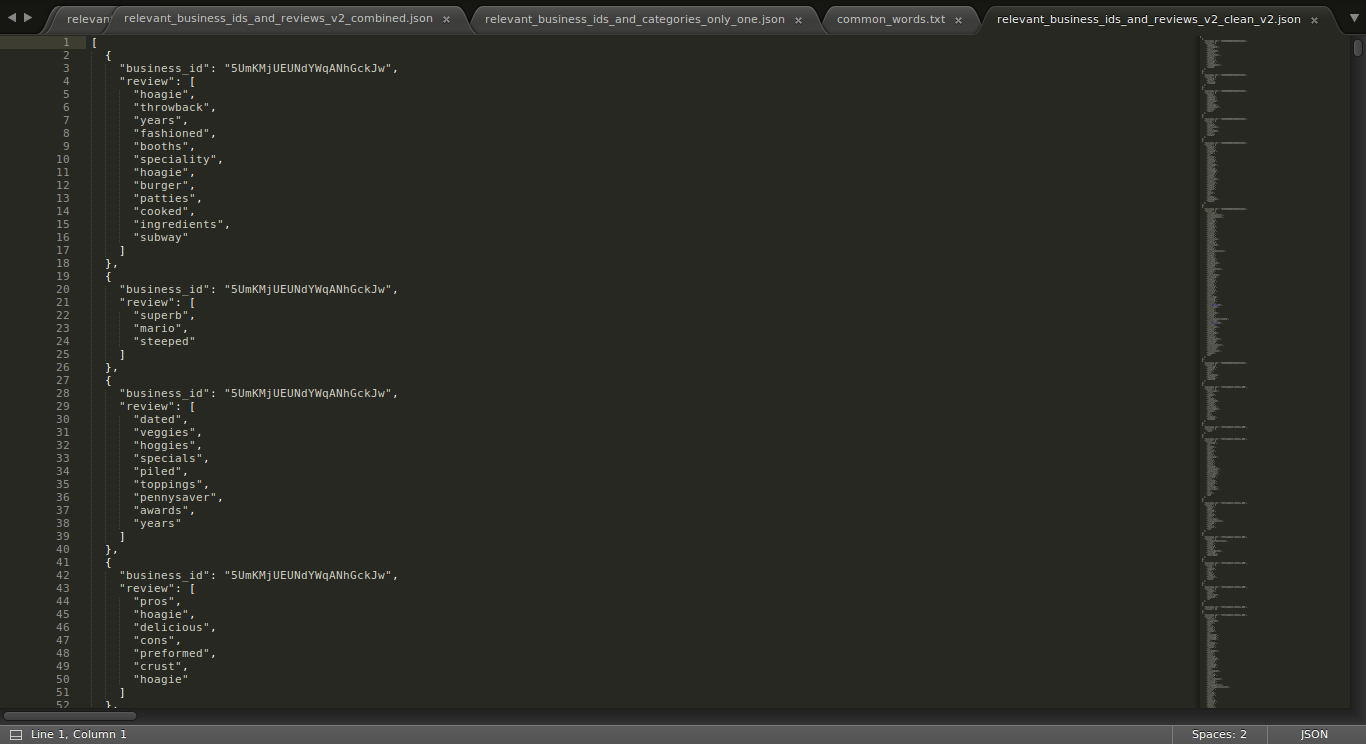
"business\_id": "KayYbHCt-RkbGcPdGOThNg"

},

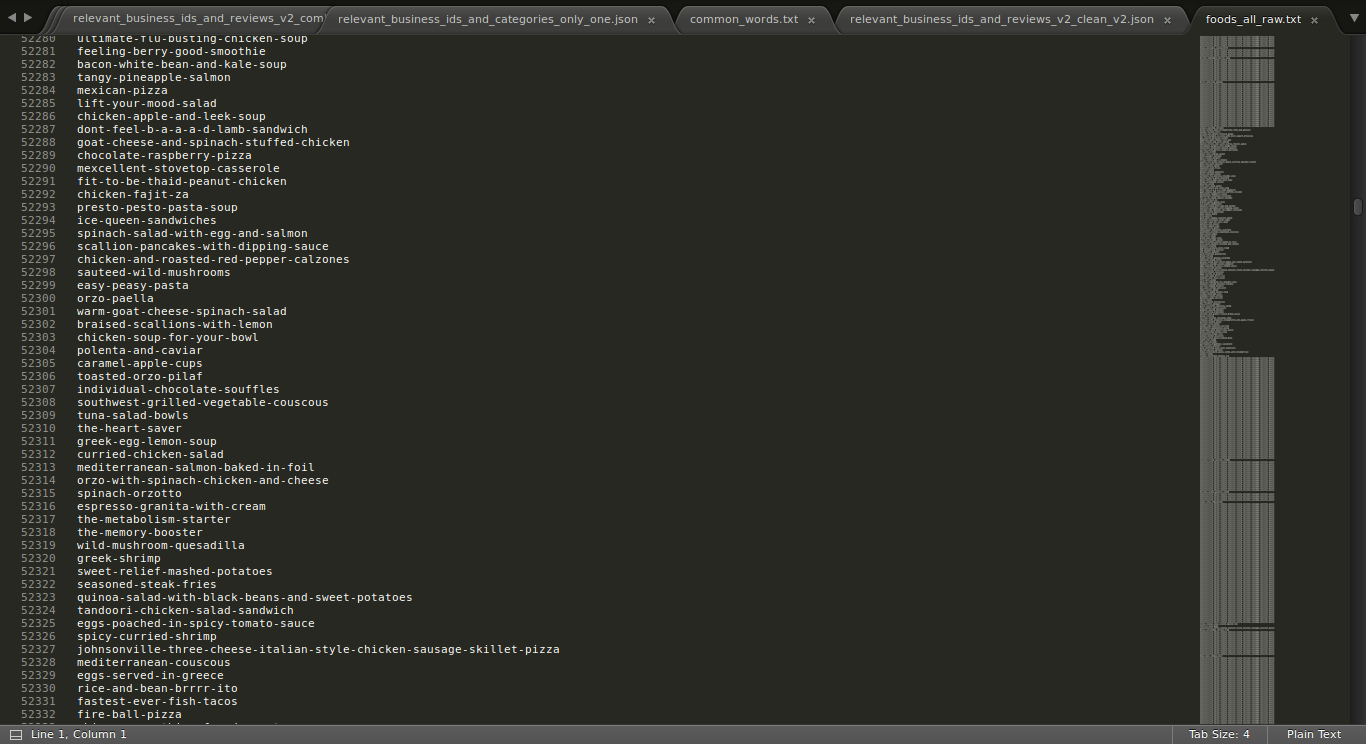
1. Now, we clean our reviews file and keep it ready to be used further in the process.
   * For this, there would be two approaches. The first approach removes all the common words, stop words, numbers, symbols and converts all reviews into lowercase words, so that comparison is uniform.
   * We have 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



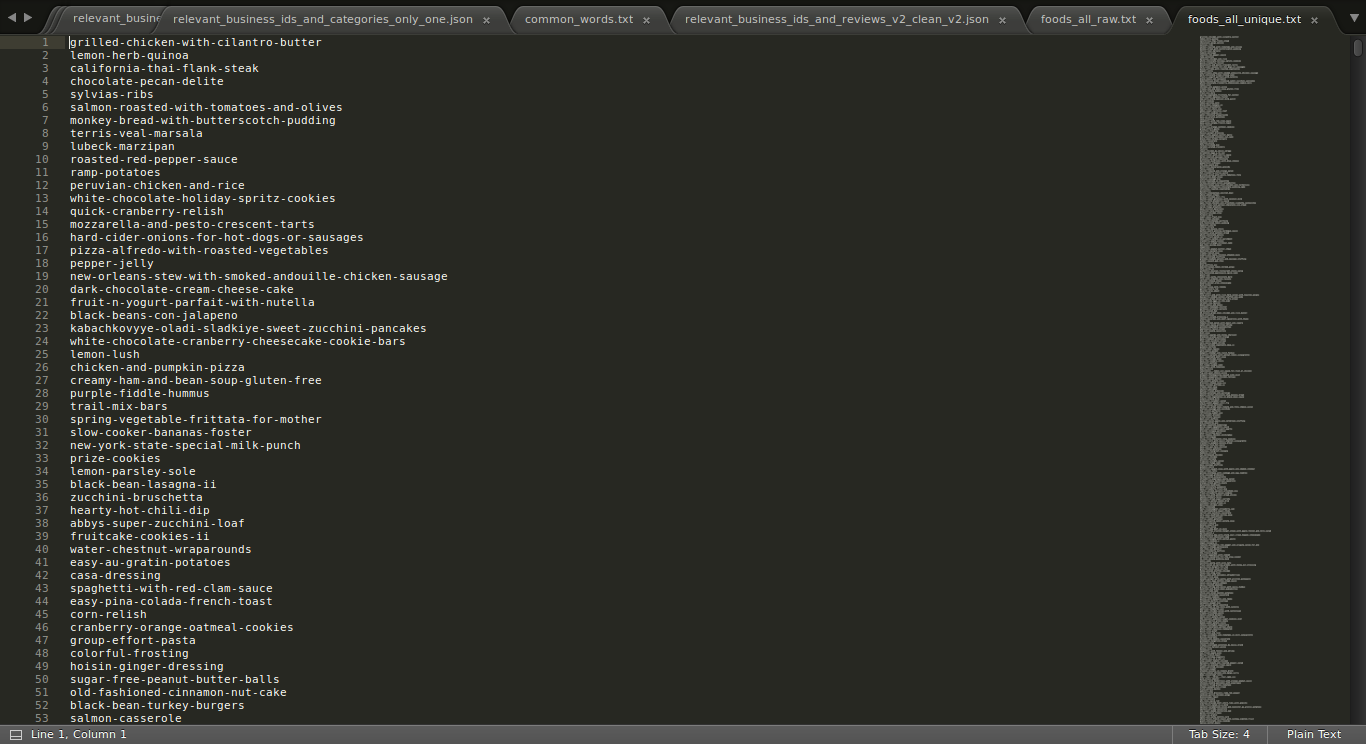
* + cleansing/relevant\_bids\_reviews\_clean.py
    - Input: **relevant\_business\_ids\_and\_reviews.json**
    - Output: **relevant\_business\_ids\_and\_reviews\_clean.json**



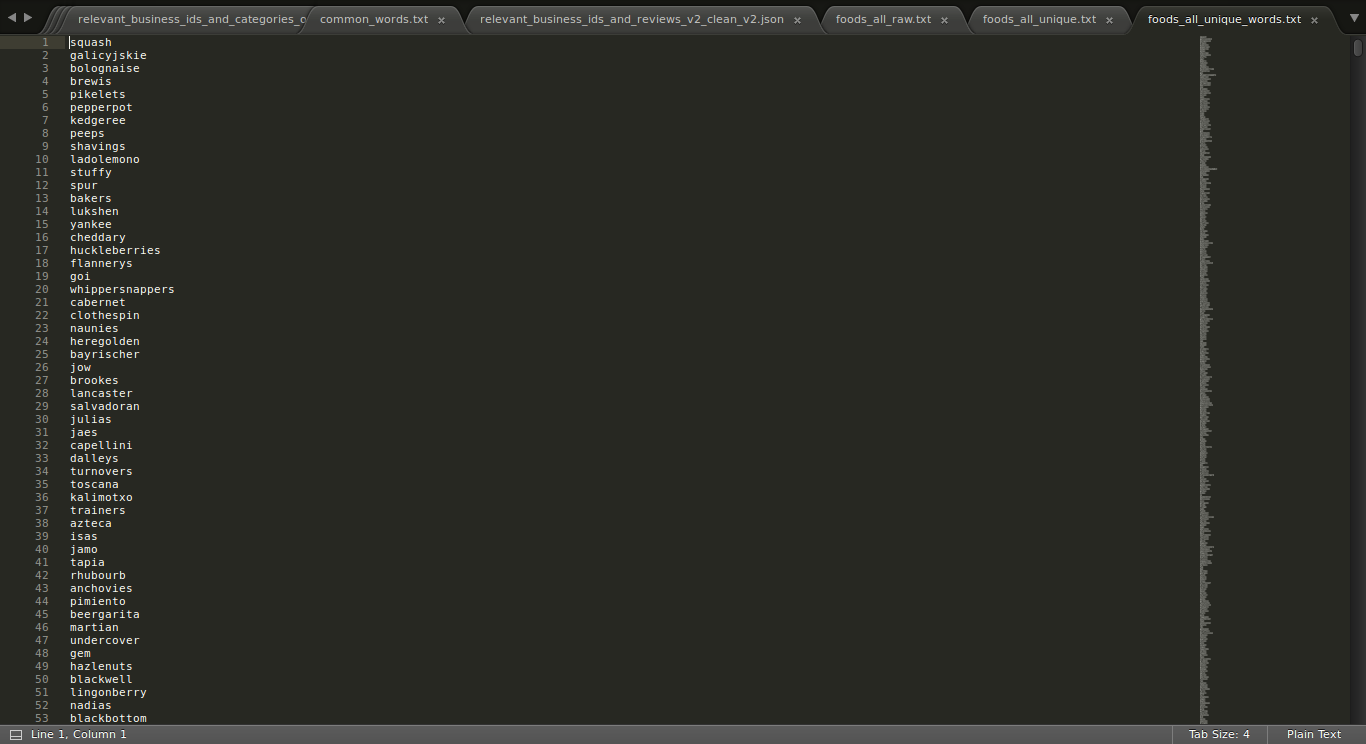
* + For the second approach, we only keep food related words in our reviews. To gather food related words, we scrapped the website <http://allrecipes.com/> for all the world’s recipies.
  + scrap/food\_scrapper.py
    - The script scrap the website and pings about 300,000 different URLs to get all the recipes.
    - Output: **foods\_all\_raw.txt**



* + - 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**

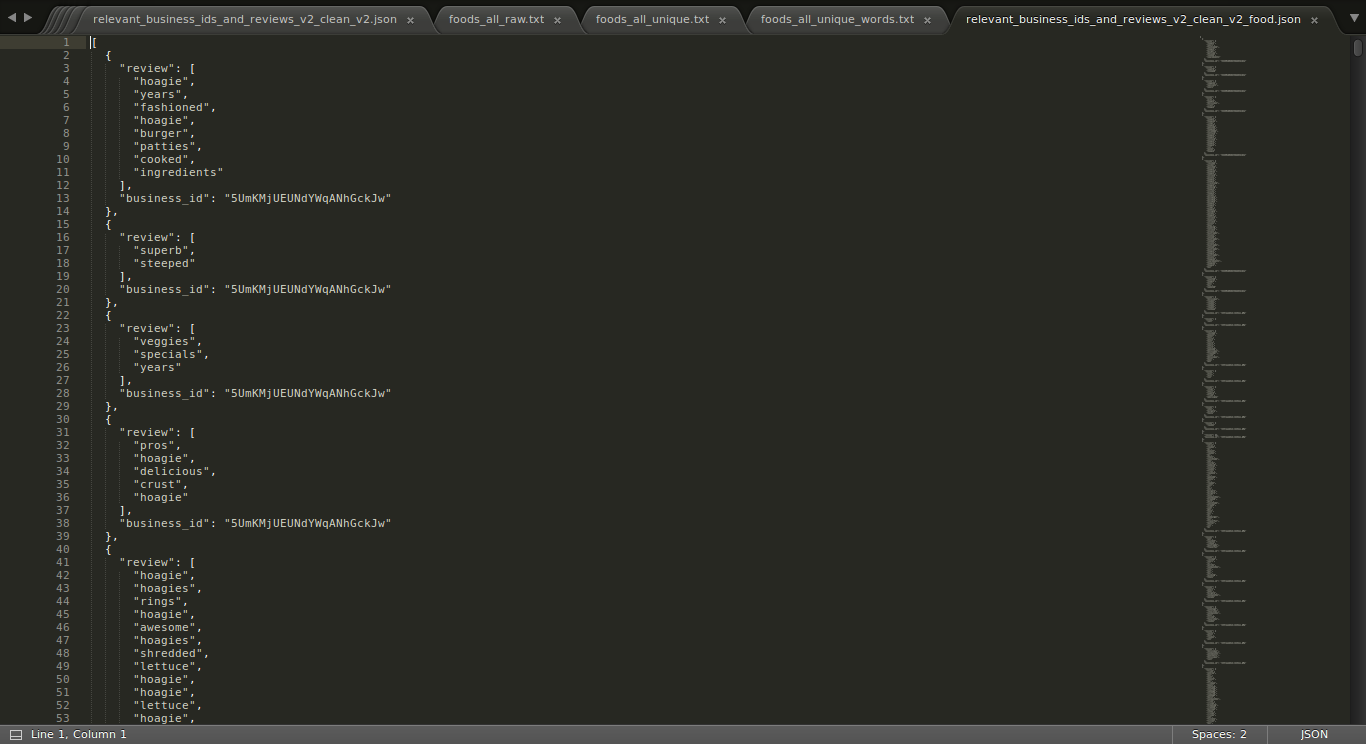


* + - * We got 77,830 unique recipes
    - Next step is to extract only food words out of these recipes. We remove common words and stopwords.
    - cleansing/food\_words\_unique\_words.py
      * Input: **foods\_all\_raw\_unique.txt**
      * Output: **foods\_all\_raw\_words.txt**



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

1. 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. 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

1. 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

1. 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.