Information Retrieval Analysis Group Project Report

Search Engine Design

Course Instructor

Prof. Cong Gao

Submitted by

Zhang Bojun
Zhang Shizhe
Wang Qingguo

Table of Contents

1. Yelp Dataset Pre - Processing

The Yelp data contains a total of six datasets, the datasets include information that Yelp obtained on business, reviews, users, checkins, tips and photos. For this search engine design project, only business, review and tip are considered useful and relevant to build the index, the table below is a table structure overview of these three datasets.

Column	Business.json	Review.json	Tip.json
1	**business_id**	review_id	**text**
2	**name**	user_id	date
3	**address**	business_id	compliment_count
4	**city**	stars	business_id
5	**state**	date	user_id
6	**postal_code**	**text**	
7	**latitude**	useful	
8	**longitude**	funny	
9	**stars**	cool	
10	review_count		
11	is_open		
12	attributes		
13	**categories**		
14	hours		

Not all columns of the datasets are necessary while building the index, those highlighted in bold red in the table above are considered as relevant columns for this project. We need to merge these three datasets before passing to Lucene to process, the merge here does not represent a inner join between datasets based on common column (business_id), as the single business entity (business_id, each line in business.json) is considered as one document of the corpus, we have to concatenate both reviews and tips under each business entity first, and then join back to business dataset, basically, the cardinality of join will become 1:1 after concatenation. We have written a simple python

script to clean and join the datasets, key steps are summarised below with code snippets, source code can be found in the submission together with this report. First, filter out the closed business entities, drop those irrelevant columns and select 50,000 out of business entities for indexing.

```
df = pd.read_json(business_json_path, lines=True)
df_open = df[df['is_open']==1]
df_open.drop(columns = ['hours','is_open','review_count','attributes'], inplace = True)
df_sample = df_open.sample(n = 50000, random_state = 2022)
df_sample.reset_index(drop=True,inplace=True)
df_sample
```

executed in 2.80s, finished 13:19:08 2022-04-10

	business_id	name	address	city	state	postal_code	latitude	longitude	stars	categories
0	OV0yJI81BkmyDu9-Om4TlQ	Founders Park	11675 Hazel Dell Pkwy	Carmel	IN	46032	39.959046	-86.072274	4.5	Parks, Active Life
1	qUfSXzTQhNms1qIsPBSQrg	Dollar Tree	6701 Market St	Upper Darby	PA	19082	39.962659	-75.255426	2.5	Discount Store, Shopping
2	oljqbMAx97IaDybrflcRSA	Arbys	5115 N 300th E	Whiteland	IN	46184	39.551317	-86.044516	2.0	Food, Fast Food, Restaurants, Beverage Store, ...

Next, concatenate reviews and tips under each business entity and save it in a dictionary whose keys are business_ids.

```
# tip dictionary
def load_tips_to_dict(t_path,tip_dict):
    with open(t_path, encoding='utf-8') as infile:
        tips = infile.readlines()
        for t in tips:
            data = json.loads(t)
            if data["business_id"] in tip_dict.keys():
                tip_dict[data["business_id"]].append(data["text"])

    return tip_dict

load_tips_to_dict(tip_json_path,tip_dict)
tip_dict
```

executed in 4.00s, finished 13:19:18 2022-04-10

```
{'OV0yJI81BkmyDu9-Om4TlQ': ['The Carmel winter farmers market is here at Founders Park Saturdays from 9 a.m. to 12 p.
m.'],
 'qUfSXzTQhNms1qIsPBSQrg': [],
 'oljqbMAx97IaDybrflcRSA': ['The roast beef and the curly fries were excellent.',
  "Attached to Love's gas station and there is a drive thru even. This is right off the expressway and in a great locatio
n for FAST food on the go! Cheers! Happy Independence Day everyone!"],
```

Finally, join review and tip back to the business dataset through mapping from keys in the review and tip dictionary to the business_id in the business dataset.

business_id	name	address	city	state	postal_code	latitude	longitude	stars	categories	tip	review
OV0yJI81BkmyDu9-Om4TlQ	Founders Park	11675 Hazel Dell Pkwy	Carmel	IN	46032	39.959046	-86.072274	4.5	Parks, Active Life	['The Carmel winter farmers market is here at ...	['The Carmel winter farmers market is here at ...
qUfSXzTQhNms1qIsPBSQrg	Dollar Tree	6701 Market St	Upper Darby	PA	19082	39.962659	-75.255426	2.5	Discount Store, Shopping	[]	['This dollar tree never have nothing always o...
oljqbMAx97IaDybrflcRSA	Arbys	5115 N 300th E	Whiteland	IN	46184	39.551317	-86.044516	2.0	Food, Fast Food, Restaurants, Beverage Store, ...	['The roast beef and the curly fries were exce...	['The roast beef and the curly fries were exce...
I7Uj583oxTXdbmqAEfCtaw	DeLisi Dental	1707 W Passyunk Ave	Philadelphia	PA	19145	39.924723	-75.174903	5.0	Health & Medical, Cosmetic Dentists, Dentists,...	['Everyone is so kind and knowledgeable! They ...	['Everyone is so kind and knowledgeable! They ...
ECxsb2wf9J7Qf2CWtJelbw	Phila Nail Wholesale Supply	571 Adams Ave	Philadelphia	PA	19120	40.037292	-75.107970	2.0	Beauty & Spas, Cosmetics & Beauty Supply, Shop...	['Terrible customer service']	['Terrible customer service', "Terrible custom...

2. Lucene Index Creation

2.1. Field and Index Creation

After importing the pre - processed Json document using Scanner to the Java coding environment, the first step is to create fields of the documents, we have classified the fields into 4 categories to serve different search purposes.

a. Tokenize text data for boolean, term query, etc
b. Create LatLonPoint for the geospatial query
c. Tokenize the numerical data for the range query and meanwhile store the field by using StoredField
d. No tokenization required and usually checked by direct match (phase query)

We select the StandardAnalyzer as the analyzer used to tokenize the input text, the index is then created and saved to the local disk, whenever there is a search query, it will go to this directory and search for the results.

```
        // Tokenize text data for boolean query, term query, etc
        doc.add(new TextField( name: "name", name, Field.Store.YES));
        doc.add(new TextField( name: "categories", categories, Field.Store.YES));
        doc.add(new TextField( name: "address", address, Field.Store.YES));
        doc.add(new TextField( name: "tip", tip, Field.Store.YES));
        doc.add(new TextField( name: "review", review, Field.Store.YES));

        // Create LatLonPoint for the geospatial query
        doc.add(new LatLonPoint( name: "latlon", latitude, longitude));

        // Tokenize the numerical data for the range query and meanwhile store the field by using StoredField
        doc.add(new DoublePoint( name: "latitude", latitude));
        doc.add(new StoredField( name: "latitude", latitude));
        doc.add(new DoublePoint( name: "longitude", longitude));
        doc.add(new StoredField( name: "longitude", longitude));
        doc.add(new DoublePoint( name: "stars", stars));
        doc.add(new StoredField( name: "stars", stars));

        // No tokenization required and usually checked by direct match
        doc.add(new StringField( name: "postal", postal, Field.Store.YES));
        doc.add(new StringField( name: "city", city, Field.Store.YES));
        doc.add(new StringField( name: "state", state, Field.Store.YES));

        docList.add(doc);

    }catch(Exception e){}
}
Analyzer analyzer = new StandardAnalyzer();
```

2.2. First Query Search

We use the default retrieval model which is BM25, only Name and Categories Fields are considered for the search. Our first boolean query search

is Breakfast AND Cafe AND Brunch, we will retrieve the top 20 (top 5 shown as an example to illustrate the idea in the report, detailed ranking list can be found using source code) ranking results together with the name, stars and categories field information.

```
@Test
public void booleanQuery() throws Exception {
    Analyzer analyzer = new StandardAnalyzer();
    String[] fields = {"name", "categories"};
    MultiFieldQueryParser multiQueryParser = new MultiFieldQueryParser(fields, analyzer);
    Query query = multiQueryParser.parse( query: "Breakfast AND Cafe AND Brunch");
    printResult(query);
}
```

```
private void printResult(Query query) throws Exception {

    Formatter formatter = new SimpleHTMLFormatter( preTag: "<font color='red'>", postTag: "</font>");
    Scorer fragmentscorer = new QueryScorer(query);
    Highlighter highlighter = new Highlighter(formatter, fragmentscorer);
    Fragmenter fragmenter = new SimpleFragmenter( fragmentSize: 500);
    highlighter.setTextFragmenter(fragmenter);

    TopDocs topDocs = indexsearcher.search(query,  n: 20);

    System.out.println("Total Records:" + topDocs.totalHits + "\n");
    ScoreDoc[] scoreDocs = topDocs.scoreDocs;

    if (scoreDocs != null) {
```

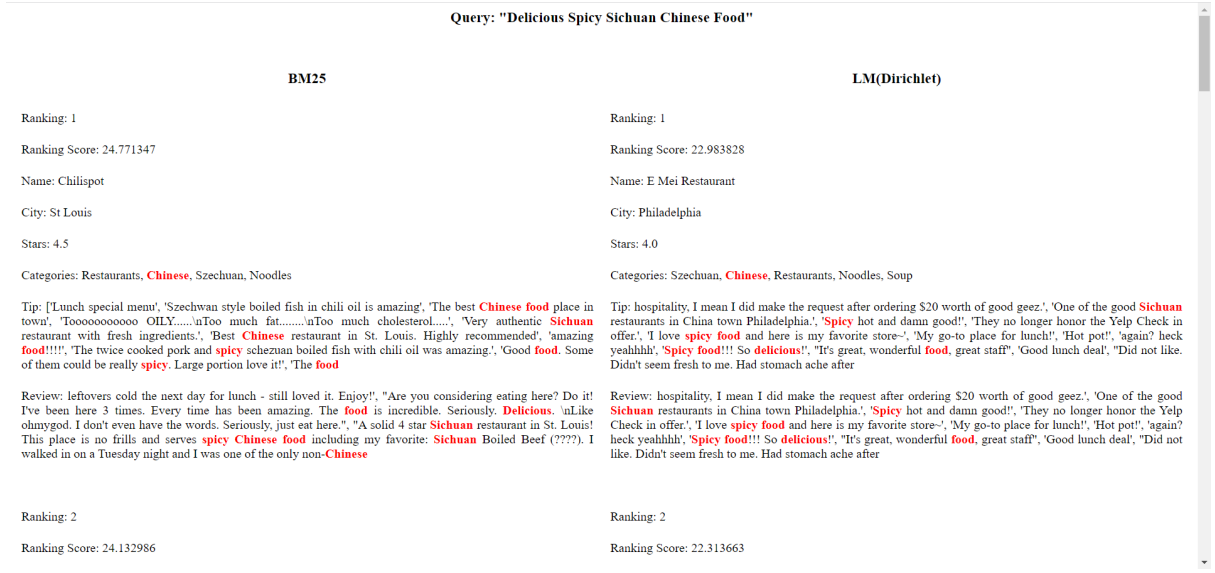
```
Ranking:1
Ranking Score:8.396789
name:Keke's Breakfast Cafe
stars:3.5
categories:Diners, Breakfast & Brunch, Restaurants, Cafes

Ranking:2
Ranking Score:8.396789
name:Keke's Breakfast Cafe
stars:4.0
categories:Breakfast & Brunch, Cafes, Diners, Restaurants

Ranking:3
Ranking Score:8.192932
name:Keke's Breakfast Cafe
stars:4.0
categories:Diners, American (Traditional), Breakfast & Brunch, Restaurants

Ranking:4
Ranking Score:8.012096
name:Keke's Breakfast Cafe
stars:3.5
categories:Breakfast & Brunch, Diners, Restaurants, Waffles, American (Traditional)

Ranking:5
Ranking Score:7.7967224
name:Buendia Breakfast & Lunch Cafe
stars:5.0
categories:Mexican, Cafes, Restaurants, Breakfast & Brunch
```

3. Experiments & Evaluation

For the experiment, BM25 and LM(Dirichlet) are the two retrieval models we choose, 20 queries are applied on these two models and make a comparison.

For the model evaluation, we use NDCG@10, basically, we judge the relevance of the ranking results of each model on a 0 - 3 relevance scale, our judgement mainly depends on the text content of name, city, categories, tip, review as well as the value of stars rating. We use one query as an example to illustrate the idea, and the same approach will be applied to the rest.

For the query, “Delicious Spicy Sichuan Chinese Food”, we compare the ranking results of two retrieval models side by side in html, keywords from the query are highlighted, since the tip and review section contain long sentences, the contents are fragmented with maximum of 500 words to display, and these fragments are considered the most relevant abstracts which include the most number of highlighted words, and then we judge the performance of the models based on above mentioned criteria.



Query: "Delicious Spicy Sichuan Chinese Food"

BM25	LM(Dirichlet)
Ranking: 1	Ranking: 1
Ranking Score: 24.771347	Ranking Score: 22.983828
Name: Chilispot	Name: E Mei Restaurant
City: St Louis	City: Philadelphia
Stars: 4.5	Stars: 4.0
Categories: Restaurants, **Chinese**, Szechuan, Noodles	Categories: Szechuan, **Chinese**, Restaurants, Noodles, Soup
Tip: ['Lunch special menu', 'Szechwan style boiled fish in chili oil is amazing', 'The best **Chinese food** place in town', 'Tooooooooooo OILY.....\nToo much fat.....\nToo much cholesterol.....', 'Very authentic **Sichuan** restaurant with fresh ingredients.', 'Best **Chinese** restaurant in St. Louis. Highly recommended', 'amazing **food**!!!!', 'The twice cooked pork and **spicy** schezuan boiled fish with chili oil was amazing.', 'Good **food**. Some of them could be really **spicy**. Large portion love it!', 'The **food**	Tip: hospitality, I mean I did make the request after ordering $20 worth of good geez.', 'One of the good **Sichuan** restaurants in China town Philadelphia.', '**Spicy** hot and damn good!', 'They no longer honor the Yelp Check in offer.', 'I love **spicy food** and here is my favorite store~', 'My go-to place for lunch!', 'Hot pot!', 'again? heck yeahhhh', '**Spicy food**!!! So **delicious**!", "It's great, wonderful **food**, great staff", 'Good lunch deal', "Did not like. Didn't seem fresh to me. Had stomach ache after
Review: leftovers cold the next day for lunch - still loved it. Enjoy!", "Are you considering eating here? Do it! I've been here 3 times. Every time has been amazing. The **food** is incredible. Seriously. **Delicious**. \nLike ohmygod. I don't even have the words. Seriously, just eat here.", "A solid 4 star **Sichuan** restaurant in St. Louis! This place is no frills and serves **spicy Chinese food** including my favorite: **Sichuan** Boiled Beef (????). I walked in on a Tuesday night and I was one of the only non-**Chinese**	Review: hospitality, I mean I did make the request after ordering $20 worth of good geez.', 'One of the good **Sichuan** restaurants in China town Philadelphia.', '**Spicy** hot and damn good!', 'They no longer honor the Yelp Check in offer.', 'I love **spicy food** and here is my favorite store~', 'My go-to place for lunch!', 'Hot pot!', 'again? heck yeahhhh', '**Spicy food**!!! So **delicious**!", "It's great, wonderful **food**, great staff", 'Good lunch deal', "Did not like. Didn't seem fresh to me. Had stomach ache after
Ranking: 2	Ranking: 2
Ranking Score: 24.132986	Ranking Score: 22.313663

```
def ndcg10(relevance_score):
    start_actual = relevance_score[0]
    for i in range(1,10,1):
        start_actual += relevance_score[i]/math.log2(i+1)
    relevance_score.sort(reverse = True)
    start_idea = relevance_score[0]
    for j in range(1,10,1):
        start_idea += relevance_score[j]/math.log2(j+1)
    return round(start_actual/start_idea,3)
```

executed in 11ms, finished 00:43:32 2022-04-17

```
# query 1: Delicious Spicy Sichuan Chinese Food
bm25_relevance_score = [3,2,2,1,3,1,2,2,2,2]
lm_relevance_score = [2,3,3,2,2,1,2,2,2,2]
print("bm25: ", ndcg10(bm25_relevance_score))
print("LM: ", ndcg10(lm_relevance_score))
```

executed in 7ms, finished 01:41:42 2022-04-17

```
bm25:  0.929
LM:  0.963
```

Query	NDCG@10	
	BM25	LM(Dirichlet)
1. Delicious Spicy Sichuan Chinese Food	0.929	0.963
2. Good Taste Ice Cream and Yogurt	0.831	0.952
3. Good Hair Salons in Upper Darby	0.745	0.393
4. Breakfast AND Cafe AND Brunch	0.92	0.941
5. Shopping Grocery with Discount	0.942	0.888
6. Fast Food AND Sandwiches	0.783	0.885
7. Dentists in Philadelphia	0.893	0.85
8. Tasty Italian Food in Philadelphia	0.972	0.972
9. Animal Shelters AND Pet Shops	0.86	0.858
10. Mexican Food OR Italian Food	0.816	0.967
11. name:Chi* AND Restaurant	0.838	0.817
12. name: Chinese~2 AND Hot Pot	0.888	0.906
13. categories: Hardware Stores - categories: Home Services	0.815	0.815
14. Best Shopping in Glen Mills	0.471	0.508
15. Vegetarian AND Thai Restaurants	0.779	0.82
16. categories:Pizza AND categories:Sandwiches AND Good	0.95	0.873
17. tip:"Good place" OR review:"Good place" AND Food	0.823	0.982
18. categories:"Skin Care" AND Beauty	0.877	0.855
19. categories:"Tobacco Shops" AND Shopping	0.948	0.931
20. Best Automotive Repair	0.786	0.789
Average	0.843	0.848

Detailed calculations of all 20 queries can be found under the Experiment progress folder.

4. Improvement of Model Performance

Model performance can be improved by implementing some techniques, one way is to boost the query by adding different weights to different fields respectively. Same words appearing in different fields are likely to have different levels of importance. Thus, by setting the weights properly, we will get more satisfying results.

The MultiFieldQueryParser class has this implementation.

```
public class MultiFieldQueryParser extends QueryParser {
    protected String[] fields;
    protected Map<String, Float> boosts;

    public MultiFieldQueryParser(String[] fields, Analyzer analyzer, Map<String, Float> boosts) {
        this(fields, analyzer);
        this.boosts = boosts;
    }

    public MultiFieldQueryParser(String[] fields, Analyzer analyzer) {
        super((String)null, analyzer);
        this.fields = fields;
    }
```

We can use a HashMap to receive the weight parameters and assign weights to different fields properly.

```
@Test
public void QueryBoost() throws Exception{
    Analyzer analyzer = new StandardAnalyzer();
    String[] fields = {"name", "categories", "tip", "review"};
    // set parameters related to relevance ranking
    HashMap<String, Float> boost = new HashMap<>();
    boost.put("categories", 10000f);
    MultiFieldQueryParser multiFieldQueryParser = new MultiFieldQueryParser(fields, analyzer, boost);
    Query query = multiFieldQueryParser.parse( query: "breakfast");
    printResult(query);

}
```

5. Document Clustering

In the pre - processing part, we randomly select 50,000 records. In order to further speed up our experiment. We shortlist the data into 10,000 records.

```
def sampling(records, num=10000):
    return records[:num]


business_ids_sample = sampling(business_ids)
names_sample = sampling(names)
categories_sample = sampling(categories)
tips_sample = sampling(tips)
reviews_sample = sampling(reviews)
```

We use nltk with some hand written rules to generate corpus for the clustering model.

```
import nltk
from nltk.text import TextCollection
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
# nltk.download('stopwords')
```

```
def generate_corpus(records):

    stop_words = set(stopwords.words('english'))
    # hand writen rules
    stop_words.add("'s")
    stop_words.add('&')
    stop_words.add('-')
    stop_words.add("'")
    stop_words.add("‘")
    stop_words.add(',')


    filtered_sentences = []
    for record in records:
        word_tokens = word_tokenize(record)
        filtered_sentence = [w.lower() for w in word_tokens if w.lower() not in stop_words]
        filtered_sentences.append(' '.join(filtered_sentence))
#     filtered_sentences_sample = filtered_sentences[:10000]

    return filtered_sentences
```

Using scikit-learn to compute TF-IDF for each document and do the clustering.

```
def computing_tfidf(sample):
    vectorizer = CountVectorizer(stop_words='english')
    tf_idf_transformer = TfidfTransformer()
    tf_idf = tf_idf_transformer.fit_transform(vectorizer.fit_transform(tips_sample))
    term_weight = tf_idf.toarray()
    return term_weight
```

There are several known issues with ‘english’, the built-in stop word list in sklearn (https://aclanthology.org/W18-2502/), but we have already done the filtering in nltk, so it won't be a problem.

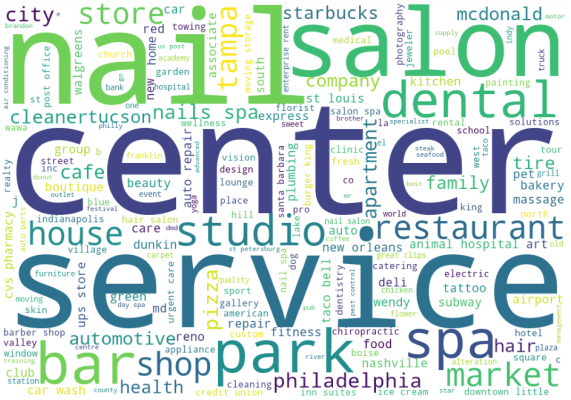
As different fields play different roles in searching, here we choose to build corpus and compute term weights for `**name**`, `**category**`, `**tip**` and`**review**` respectively.

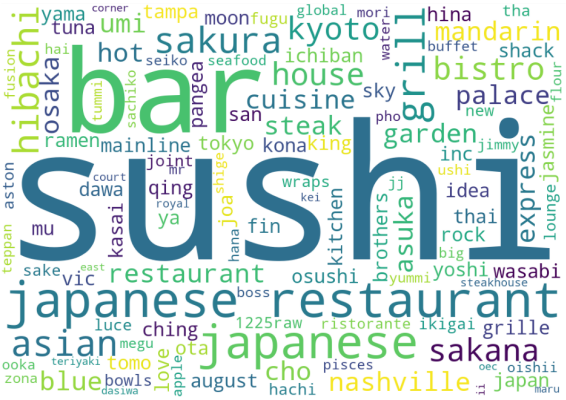
```
def culster_word_cloud(k, term_weight, corpus):
    kmeans = KMeans(n_clusters=k, random_state=0).fit(term_weight)
    labels = list(kmeans.labels_)
    for label in list(set(labels)):
        result = []
        for i in range(len(labels)):
            if labels[i] == label:
                result.append(corpus[i])
        plot_word_cloud(result)
```

We choose k-means to model the model. Based on the experimental results, setting k = 10, can give us relatively better results. Finally results are plotted via word clouds.

```
def plot_word_cloud(result):

    wc = WordCloud(
            background_color='white',
            width=1000,
            height=700,
            max_font_size=500,
            min_font_size=10,
            mode='RGBA'
            #colormap='pink'
            )

    wc.generate(' '.join(result))

    #wc.to_file(r"wordcloud.png")

    # plt.figure("mywordcloud")
    plt.figure(figsize=(18, 12))
    plt.imshow(wc)
    plt.axis("off")
    plt.show()
```

`name`






pub
british
fox
hounds


leisure
centre
kinsmen

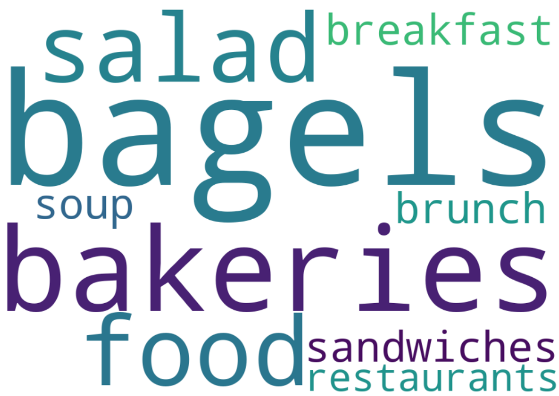

coffee
starbucks
cafe
house
dunkin
roasters
company
truck
mcdonald
village
tucson
market
bakery
café

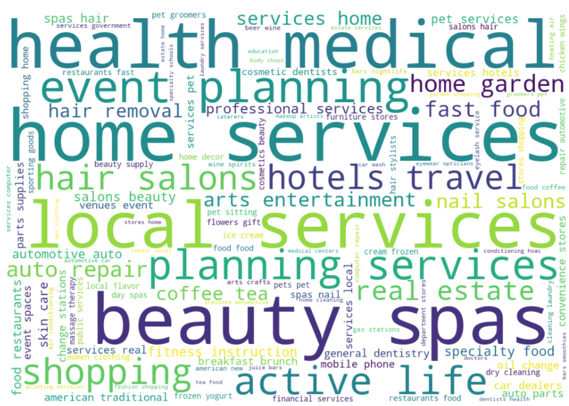

pizza
pizzeria
restaurant
pasta
domino
tony
grill
italian
chicago
new york
little caesar

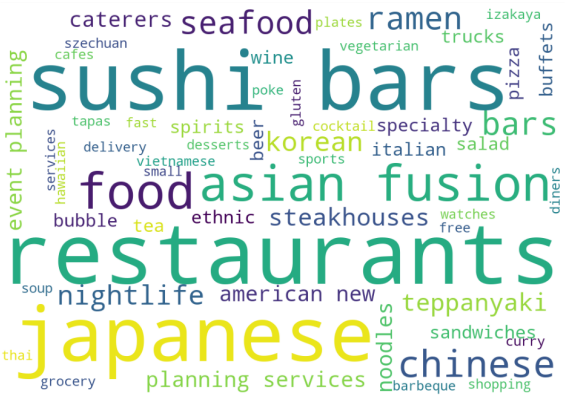

restaurant
kitchen
bar
house
grill
cafe
pizza
italian
taco
deli
nail
pub
thai
diner
tavern
food
seafood
mexican

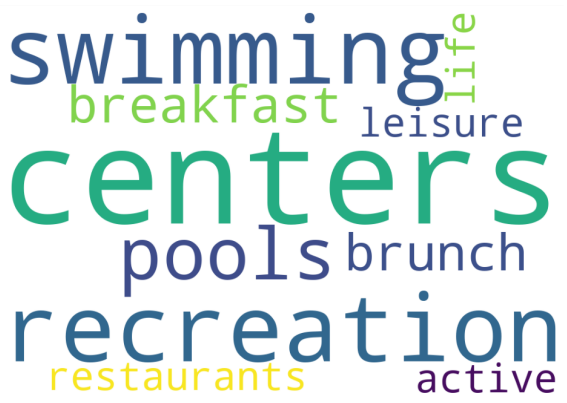

hospital
banfield
pet

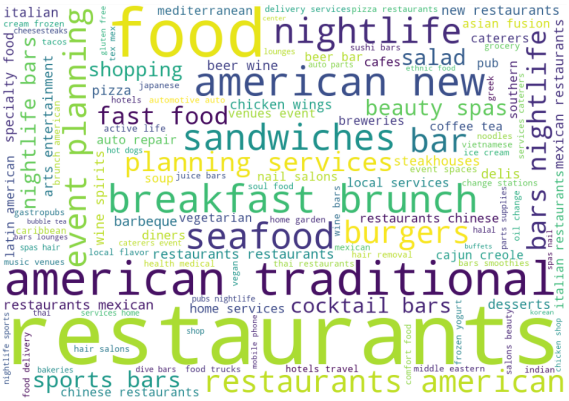
`category`


salad
breakfast
bagels
soup
brunch
bakeries
food
sandwiches
restaurants


health medical
event planning
home services
hair salons
hotels travel
local services
planning services
beauty spas
shopping
active life
real estate
auto repair



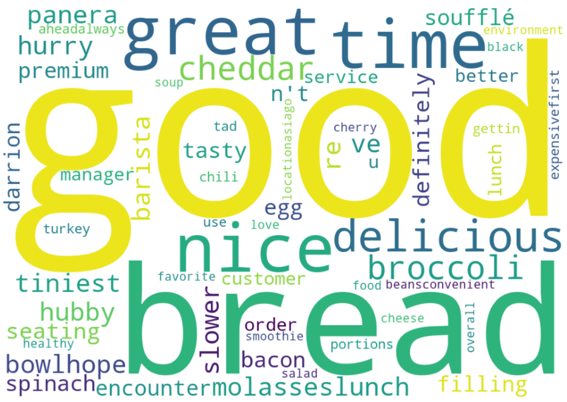




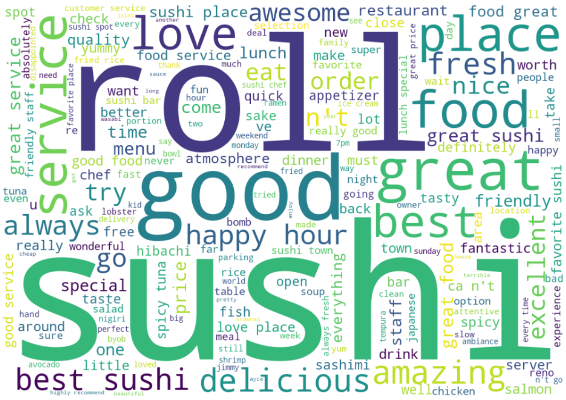
pets

veterinarians

`tip`

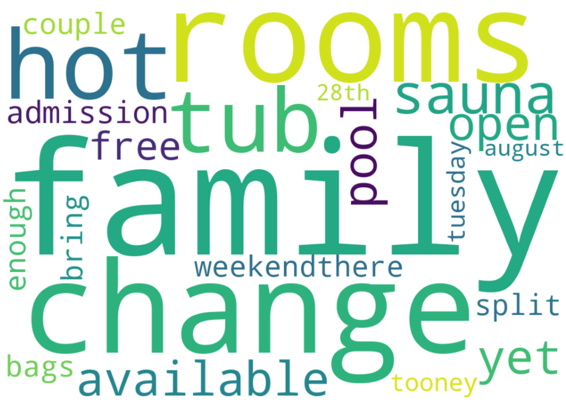

good
bread
great
time
nice
delicious
broccoli
cheddar
panera
hurry
premium
soufflé
tasty
barista
darrion
manager
turkey
tiniest
hubby
seating
bowlhope
spinach
encounter
molasses
lunch
slower
order
bacon
portions
filling
definitely
better
egg
cherry


great
place
best
good
love
time
service
awesome
customer service
need
one
come
try
ice cream
check
location
food
nice
new
amazing
store
day
clean
drink
price
shop
friendly
staff


sushi
roll
good
great
service
best
love
place
fresh
food
nice
order
always
happy hour
best sushi
delicious
amazing
excellent
great sushi
try
sushi place
awesome
restaurant


ave
service
go
good
one
cottman


food
cool
good
n't
dive
wall
chill
open
woohoo
great
till
brandon
hole
selection
bartenders
beer
still
oldest
pub
weekend
little
going
came
seem
much
lunch
menu
bar
smoking
spot
tap
allow
served
limited
inside


family
change
rooms
hot
tub
sauna
open
august
couple
admission
free
28th
pool
tuesday
bring
enough
weekend
there
split
bags
available
tooney
yet

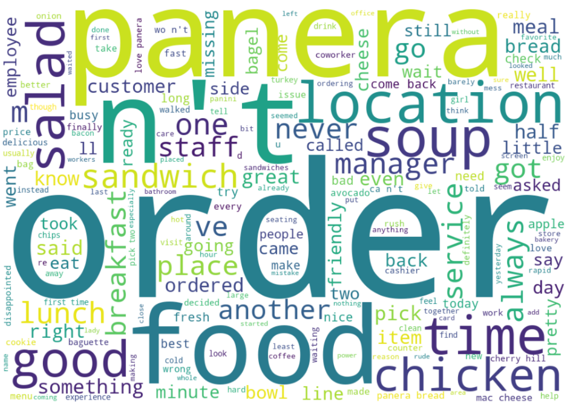

coffee
great
place
good
best
food
delicious
service
breakfast
nice
love
latte
drink
try
one
n't
amazing
always
people
friendly
great coffee
irish coffee
starbucks
excellent
iced coffee


pizza
place
great
good
food
best
great pizza
excellent
atmosphere
favorite
delicious
amazing
awesome
service
good pizza
love
pepperoni
cheese
crust
slice
wings
order
fast
local


good
best
great food
place
love
great
delicious


love
becky
dr

`review`


panera
order
food
n't
location
soup
chicken
time
good
salad


one
ca n't
customer service
ice cream
well
got
work
store


roll
one
place
restaurant
table
well
good
food
ordered
menu


wing
order
n't
food
boneless
fries
place
service
time


great
place
food
drink
smoke
bar
good
beer
wing
bartender
pub
inside
service
selection
tracy
ordered
brandon
tampa
never
staff
smoking
wanted
love
one
n't
indoor smoking
local
friendly
people
smoker
popper
jalapeno
always
come
attentive
time
amazing
trying
remember
delicious
bathroom
hipsters
cool
cheese
menu
walking
strong
find
night
scotty
hole
better
college
came
pretty
new
hate
close
dive
pool
water
swimming
n't
one
staff
party
area
space
room
park
kid
really
place
family
choose
change
play
front
use
better
many
users
old
ca
never
stairs
lessons
facility
door
small
building
even
millennium
birthday
lot
much
children
aquasize
upstairs
parking
enjoy
spots
grandma
ve
pretty
stalls
toddler
shallow
getting
energy
counter
floor
slide
perfect
instructor
kinsmen
seen
lifeguard
bathroom
lady
allowed
behind
yelled
commute
complaint
lost
hour
climbing
next
limit
dirty
tried
little
something
close
park
used
millenium
decided
kiddos
go
take
think
great
clean
swim
crowded
minutes
forced
class
main
say
walk
way
sportball
hot
salt
decent
older
usually
different
mistake
aquasize
children
mostly
eat
pretty
local
lobby
much
deep
far
big
reception
item
venue
unwelcome
everyone
maintained definitely
food
coffee
place
one
good
drink
n't
ordered
love
great
order
nice
location
delicious
always
menu
go
got
bar
restaurant
work
still
need
amazing
option
iced
coffee
customer service
owner
bakery
starbuck
right
asked
really
experience
day
make
back
table
people
bagel
time
well
came
cake
sandwich
sweet
breakfast
donut
coffee shop
made
two
perfect
brunch
morning
tasty
small
think
bit
area
spot
everything
better
awesome
atmosphere
fresh
eat
crepe
meal
another
ca n't
taste
flavor
price
even
u
wonderful
staff
cafe
pastries
friend
thing
latte
service
review
recommend
inside
tried
first
stop
around
ve
try
said
barista
french toast
breakfast
know
sandwich
definitely
customer
sweet
see
look
little
busy
favorite
local
hour
enjoy
crepe
side
full
check
serve
pizza
place
one
good
got
restaurant
salad
food
order
well
make
cheese
pie
ordered
delicious
slice
wing
love
area
still
went
table
deep dish
people
time
flavor
tried
come
review
family
sandwich
day
kind
eating
another
special
staff
back
next time
new york
crust
location
server
bread
pizza
good pizza
great pizza
ordering
everything
option
right
eat
taste
sauce
came
menu
cheese
dough
fries
bar
pizza good
drink
amazing
topping
pizza place
need
pasta
excellent
bit
happy
style pizza
left
bad
owner
much
guy
even
thin crust
offer
ok
price
made
small
maybe
fresh
cheesesteak
delivery
really good
visit
customer service
first time
took
see
night
little
vet
dr
banfield
dog
pet
told
plan
love
staff
good
cat
time
lunch
location
day
n't
appointment
care
today
office
normally
richmond
asked
call
nice
year
wellness
doctor
highly
recommend
done
different
another
service
accommodating
patient
week
teeth
eschmann
clayton
well
waiting
everything
happy
kind
keisha
bring
take
immediately
answered
anxious
cleaned
cared
still
include
feel
support
particular
cleaning
great
made
offer
carrier
customers
pups
leaving
questions
give
need
detail
given
early
overly
case
check
incredibly
wonderful
working
people
knowledgeable
calm
home
ready
around
worked
sarah
accommodated
dr
girls
really
amanda
right
release
went
emergency
heights
total
hour
candice
came
always
concern
us
thank
hand
best
wellness
year
pickup
peaches
processed
recommended
cunningham
schedule
sick
normally
richmond
asked
call
nice
year
wellness
doctor
highly
accommodating
first
time
can
't
one
well
go
back
food
good
great
place
next time
customer service
dinner
n't know
food service
restaurant
said
wo n't
mac cheese
instead
entree
husband
either
really good
appetizer
dish
pretty good
point
meal
though
maybe
went
although
kind
location
perfect
love
owner
cocktail
mexican food
fried chicken
excellent
n't wait
burger
loved
look
amazing
put
dishe
think
work
special
service food
ve never
ve favorite
n't even
taste
flavor
wife
asked
say
side
wonderful
use
people
manager
come back
great service
food great
husband
either
way
really good
highly recommend
pretty good
meal
though
maybe
went
although
kind
location
perfect
love
owner
cocktail
mexican food
fried chicken
excellent
great place
next time
customer service
dinner
n't know
food service
huge
love place
thank
especially
definitely back
great
see
food
awesome
going back
lot
tried
thing
make
fantastic
waiter
plate
one
best
right
look
amazing
put
dishe
think
burger
loved
work
special
service food
ve never
ve favorite
n't even
taste
flavor
wife
asked
say
side
wonderful
use
people
manager
restaurant
said
wo n't
mac cheese
instead
entree
husband
either
way
really good
appetizer
dish
pretty good
point
meal
though
maybe
went
although
kind
location
perfect
love
owner
cocktail
mexican food
fried chicken
excellent
great place
next time
customer service
dinner
n't know
food service
huge
first
time
can
't
one
well
go
back
food
good
great
place
next time
customer service
dinner
n't know
food service
restaurant
said
wo n't
mac cheese
instead
entree
husband
either
really good
appetizer
dish
pretty good
point
meal
though
maybe
went
although
kind
location
perfect
love
owner
cocktail
mexican food
fried chicken
excellent
great place
next time
customer service
dinner
n't know
food service
huge

6. Geospatial Indexing & Search

To handle one type of geospatial queries like finding a restaurant in an area, we use the BooleanQuery.Builder by combining the description of the restaurant and location of the area, below is the snippet of the code implementation and the result for the query, Tasty Italian Food in Philadelphia.

```
@Test
public void geo1_combinedQuery() throws Exception{
    String[] fields = {"name","categories","tip","review"};
    MultiFieldQueryParser multiQueryParser = new MultiFieldQueryParser(fields, analyzer);
    Query query1 = multiQueryParser.parse( query: "Tasty Italian Food in Philadelphia");
    Query geo_1 = new PhraseQuery.Builder()
            .add(new Term( fld: "city",  text: "Philadelphia")).build();

    BooleanQuery.Builder booleanQuery = new BooleanQuery.Builder();
    booleanQuery.add(query1, BooleanClause.Occur.MUST);
    booleanQuery.add(geo_1, BooleanClause.Occur.MUST);
    printResult(booleanQuery.build());
}
```

```
Ranking:1
Ranking Score:15.685847
name:Rocco's Italian Sausage
city:Philadelphia
stars:4.5
categories:Sandwiches, Breakfast & Brunch, Restaurants, Cheesesteaks, Italian

Ranking:2
Ranking Score:15.257706
name:Tortorice's Italian Specialty Sandwiches
city:Philadelphia
stars:4.5
categories:Event Planning & Services, Food, Italian, Coffee & Tea, Sandwiches, Restaurants, Caterers, Delis

Ranking:3
Ranking Score:14.793793
name:Francoluigi's Pizzeria & Italian Restaurant
city:Philadelphia
stars:4.0
categories:Italian, Restaurants, Pizza

Ranking:4
Ranking Score:14.347813
name:Spasso Italian Grill
city:Philadelphia
stars:4.0
categories:Sports Bars, Restaurants, Bars, American (Traditional), Seafood, Nightlife, Italian, Beer Bar
```

To handle the other type of geospatial queries like finding the nearest restaurant, we need to create a new type of Field for each document, the LatLonPoint Field which combines latitude and longitude into one field.

```
// Create LatLonPoint for the geospatial query
doc.add(new LatLonPoint( name: "latlon", latitude, longitude));
```

Assume we know our current coordinate information, and we want to find certain restaurants within a certain distance. For instance, we want to find a tasty Italian restaurant within 1km from our current location (latitude: 40.0, longitude: -75.0), below is the snippet of the code implementation and the result for this query, details can be found from the source code.

```
@Test
public void geo2_combinedQuery() throws Exception {
    String[] fields = {"name", "categories", "tip", "review"};
    MultiFieldQueryParser multiQueryParser = new MultiFieldQueryParser(fields, analyzer);
    Query query1 = multiQueryParser.parse( query: "Tasty Italian Food in Philadelphia");
    Query geo_1 = LatLonPoint.newDistanceQuery( field: "latlon",  latitude: 40.0,  longitude: -75.0,  radiusMeters: 1 * 1000);

    BooleanQuery.Builder booleanQuery = new BooleanQuery.Builder();
    booleanQuery.add(query1, BooleanClause.Occur.MUST);
    booleanQuery.add(geo_1, BooleanClause.Occur.MUST);
    printResult(booleanQuery.build());
}
```

```
Ranking:1
Ranking Score:5.907668
name:Sonic Drive-In
city:Cinnaminson
latitude:39.9978761
longitude:-74.992041
stars:3.0
categories:Burgers, Ice Cream & Frozen Yogurt, Restaurants, Food, Fast Food

Ranking:2
Ranking Score:4.2511806
name:Cinnaminson Diner
city:Cinnaminson
latitude:39.9983826
longitude:-74.9889068
stars:3.5
categories:Breakfast & Brunch, Restaurants, Diners

Ranking:3
Ranking Score:3.9243076
name:Speed Raceway
city:Cinnaminson
latitude:39.9989782639
longitude:-74.9917004577
stars:3.5
categories:Active Life, Arts & Entertainment, Venues & Event Spaces, Axe Throwing, Arcades, Summer Camps, Party & Event Planning, Fitness
 & Instruction, Go Karts, Event Planning & Services, Virtual Reality Centers
```