Final Report – MPCS53112 Advanced Data Analytics

MapReduce and Spark in text mining

Qian Wang, Qianyu Deng

Introduction:

The aim of our project is to develop predictive models using mainly unstructured data. In practice, the size of data in text analytics is very huge. Also the computation cost of machine learning algorithms involved in text mining such as clustering techniques that are important for extracting concept features is demanding. In this project, we want to take the advantage of big data techniques such as MapReduce and Spark in text analytics.

Objective:

We want to build a text analytics system that provides: 1) Text information retrieval system, which helps to extract structured fields and also fetch complete customer reviews from Yelp (data extraction). 2) MapReduce implementation for computing term frequency-inverse document frequency (tf-idf, value) paired with its key (restaurant, term). 3) Spark (& sklearn) implementation of k-means clustering to extract text features based on tf-idf (feature extraction). 4) Classification models on predicting ratings (unbalanced labels) using both structured variables and term features.

Methods:

We use Python to implement the following functionalities:

(1) Use rauth request to fetch restaurants' information with the parameter setting of location origin and radius. Yelp responses with restaurants' information including restaurant name, rating, review count and url of its website in the format of JSON (python packages rauth, requests).

(2) Implement web crawler to fetch complete reviews using the url of different restaurants returned from step (1) (python package BeautifulSoup).

(3) Clean the reviews by removing stopwords and punctuations, and stemming (python packages nltk, re).

(4) Because Yelp has a limitation on the maximum return from a query (i.e. 20 restaurants), we repeated request the restaurant information with different parameter settings and implement file merger code to merge both review .txt files and .csv files.

(5) Use 2-pass MapReduce to output tf-idf file with each line of (restaurant, term) – tf-idf (Python package mrjob). This is run on google cloud cluster.

(6) Use the tf-idf file as the input and generate a (restaurant, term) matrix of tf-idf. Also generate a corpus file with the appeared words.

(7) Use Spark to implement K-means clustering with an input of tf-idf matrix file (Python package pyspark.mllib) and this is run on google cloud cluster.

(8) Develop rating prediction models using limited structured fields and a pool of text term features (Python package sklearn).

Results:

The following are examples of generated data from the above describe steps.

Figure1. An example of a JSON response by requesting restaurant information from Yelp. This is a dictionary data structure in Python. Interested fields include “is_claimed”, “rating”, “url” and “city”.


```
{"region": {"span": {"latitude_delta": 0.0031306000000057566, "longitude_delta":
0.0016872900000066693}, "center": {"latitude": 40.731113, "longitude": -74.00091405}},
"total": 94, "businesses": [{"is_claimed": true, "rating": 4.5, "mobile_url": "https://
m.yelp.com/biz/mighty-bowl-new-york?adjust_creative=qvtAmgqN2IgX1-
x54gTQZw&utm_campaign=yelp_api&utm_medium=api_v2_search&utm_source=qvtAmgqN2IgX1-
x54gTQZw", "rating_img_url": "https://s3-media2.fl.yelpcdn.com/assets/2/www/img/
99493c12711e/ico/stars/v1/stars_4_half.png", "review_count": 86, "name": "Mighty Bowl",
"rating_img_url_small": "https://s3-media2.fl.yelpcdn.com/assets/2/www/img/a5221e66bc70/
ico/stars/v1/stars_small_4_half.png", "url": "https://www.yelp.com/biz/mighty-bowl-new-
york?adjust_creative=qvtAmgqN2IgX1-
x54gTQZw&utm_campaign=yelp_api&utm_medium=api_v2_search&utm_source=qvtAmgqN2IgX1-
x54gTQZw", "is_closed": false, "id": "mighty-bowl-new-york", "phone": "2127775750",
"snippet_text": "I'm so in love. I look for any reason to try once and move on, especially
when it comes to lunch. Mighty Bowl is the exception. It's the reason I keep...",
"image_url": "https://s3-media3.fl.yelpcdn.com/bphoto/B2pdJPk_dW9CqayvCc-Oqg/ms.jpg",
"categories": [["Asian Fusion", "asianfusion"], ["Japanese", "japanese"], ["Salad",
"salad"]], "display_phone": "+1-212-777-5750", "rating_img_url_large": "https://s3-
media4.fl.yelpcdn.com/assets/2/www/img/9f83790ff7f6/ico/stars/v1/stars_large_4_half.png",
"menu_provider": "eat24", "distance": 167.99789380060477, "menu_date_updated": 1472898808,
"snippet_image_url": "https://s3-media4.fl.yelpcdn.com/photo/v-VMoqX_Dum1usupBEX9qA/
ms.jpg", "location": {"city": "New York", "display_address": ["120 Macdougal St",
"Greenwich Village", "New York, NY 10012"], "geo_accuracy": 8.0, "neighborhoods":
["Greenwich Village"], "postal_code": "10012", "country_code": "US", "address": ["120
Macdougal St"], "coordinate": {"latitude": 40.729793, "longitude": -74.00035},
"state_code": "NY"}}, {"is_claimed": true, "rating": 4.5, "mobile_url": "https://
m.yelp.com/biz/pommes-frites-new-york-2?adjust_creative=qvtAmgqN2IgX1-
x54gTQZw&utm_campaign=yelp_api&utm_medium=api_v2_search&utm_source=qvtAmgqN2IgX1-
```

Figure 2. An example of a piece of dataset extracted from JSON response (part of a .csv file).

3	83	Come Prima Ristorante	TRUE	28	https://www.yelp.com/biz/come-prima-ristorante-new-york-2?adjust_crea	4.5	New York
4	84	Burger One	TRUE	130	https://www.yelp.com/biz/burger-one-new-york?adjust_creative=qvtAmg	4	New York
5	85	Via Quadronno	TRUE	401	https://www.yelp.com/biz/via-quadronno-new-york?adjust_creative=qvtA	4	New York
6	86	Bluestone Lane	TRUE	162	https://www.yelp.com/biz/bluestone-lane-new-york-7?adjust_creative=qv	4	New York
7	87	Dos Toros Taqueria	TRUE	295	https://www.yelp.com/biz/dos-toros-taqueria-new-york-3?adjust_creative	3.5	New York
8	88	The New Amity Restaurar	TRUE	100	https://www.yelp.com/biz/the-new-amity-restaurant-new-york?adjust_cre	3.5	New York
9	89	HARBS - Upper East Side	TRUE	73	https://www.yelp.com/biz/harbs-upper-east-side-new-york-2?adjust_creat	3.5	New York
10	90	Sant Ambroeus	TRUE	274	https://www.yelp.com/biz/sant-ambroeus-new-york?adjust_creative=qvtA	4	New York
11	91	mamagyro	TRUE	298	https://www.yelp.com/biz/mamagyro-new-york?adjust_creative=qvtAmgq	3.5	New York
12	92	The Loeb Boathouse	TRUE	914	https://www.yelp.com/biz/the-loeb-boathouse-new-york-2?adjust_creativ	3.5	New York
13	93	Vivolo Restaurant	FALSE	129	https://www.yelp.com/biz/vivolo-restaurant-new-york-3?adjust_creative=	4	New York
14	94	Eric Kayser	TRUE	541	https://www.yelp.com/biz/eric-kayser-new-york?adjust_creative=qvtAmgq	4	New York
15	95	Eats	TRUE	312	https://www.yelp.com/biz/eats-new-york?adjust_creative=qvtAmgqN2lgX	3.5	New York
16	96	The Simone	FALSE	57	https://www.yelp.com/biz/the-simone-new-york?adjust_creative=qvtAmg	4.5	New York
17	97	Marche Madison	TRUE	22	https://www.yelp.com/biz/marche-madison-new-york-2?adjust_creative=	3.5	New York
18	98	Ristorante Morini	TRUE	135	https://www.yelp.com/biz/ristorante-morini-new-york?adjust_creative=qv	4	New York
19	99	Due	FALSE	46	https://www.yelp.com/biz/due-new-york?adjust_creative=qvtAmgqN2lgX1	4	New York
0	100	Brightwok Kitchen	TRUE	278	https://www.yelp.com/biz/brightwok-kitchen-chicago?adjust_creative=qvt	4.5	Chicago
1	101	Cafecito	TRUE	1258	https://www.yelp.com/biz/cafecito-chicago?adjust_creative=qvtAmgqN2lg	4.5	Chicago
2	102	Roanoke Restaurant	TRUE	25	https://www.yelp.com/biz/roanoke-restaurant-chicago?adjust_creative=q	4	Chicago
3	103	The Gage	TRUE	2151	https://www.yelp.com/biz/the-gage-chicago?adjust_creative=qvtAmgqN2l	4	Chicago
4	104	The Marq	TRUE	228	https://www.yelp.com/biz/the-marq-chicago-2?adjust_creative=qvtAmgq	4	Chicago
5	105	The Dearborn	TRUE	148	https://www.yelp.com/biz/the-dearborn-chicago-2?adjust_creative=qvtAm	4.5	Chicago
6	106	Remington's	TRUE	228	https://www.yelp.com/biz/remingtons-chicago?adjust_creative=qvtAmgq	4	Chicago
7	107	Osaka Sushi Express & Fre	TRUE	355	https://www.yelp.com/biz/osaka-sushi-express-and-fresh-fruit-smoothies-	4	Chicago
8	108	Nando's Peri-Peri	TRUE	151	https://www.yelp.com/biz/nandos-peri-peri-chicago-18?adjust_creative=q	4	Chicago
9	109	Sociale Chicago	TRUE	151	https://www.yelp.com/biz/sociale-chicago-chicago?adjust_creative=qvtAm	4	Chicago
10	110	Sofi Restaurant	TRUE	219	https://www.yelp.com/biz/sofi-restaurant-chicago?adjust_creative=qvtAm	4	Chicago
11	111	Ge Pa De Caffe	TRUE	265	https://www.yelp.com/biz/ge-pa-de-caffe-chicago-3?adjust_creative=qvtA	4.5	Chicago
12	112	Thai Spoon	TRUE	57	https://www.yelp.com/biz/thai-spoon-chicago-4?adjust_creative=qvtAmgq	4	Chicago
13	113	Cochon Volant	TRUE	293	https://www.yelp.com/biz/cochon-volant-chicago?adjust_creative=qvtAm	4	Chicago
14	114	Mercat a la Planxa	TRUE	1224	https://www.yelp.com/biz/mercat-a-la-planxa-chicago?adjust_creative=qv	4	Chicago
15	115	First Draft	TRUE	301	https://www.yelp.com/biz/first-draft-chicago?adjust_creative=qvtAmgqN2	4	Chicago
16	116	Oasis Cafe	FALSE	358	https://www.yelp.com/biz/oasis-cafe-chicago?adjust_creative=qvtAmgqN2	4	Chicago
17	117	Native Foods Cafe	TRUE	570	https://www.yelp.com/biz/native-foods-cafe-chicago-3?adjust_creative=q	4.5	Chicago

Figure 3. A piece of fetched review .txt file with the first position as the restaurant index and following by its reviews as one line.

```
0 I'm so in love. I look for any reason to try once and move on, especially when it comes
to lunch. Mighty Bowl is the exception. It's the reason I keep moving on from every other
not-as-perfect place. It was all worth it to have found the one, true, Asian-Fusion bowl.
My first foray into their menu, skepticism level high, I ordered the California with
chicken as the protein - avocado, sprouts, kale, mushroom, and more topped with Japanese
mayo and miso bbq... The perfectly poached egg atop a bed of fresh ingredients left my
taste buds wanting more and more, despite my overstuffed stomach at the end of the meal.
Of course, I was soon back for another, trying the Tokyo next - chicken with picked
shitake, veggies, and smoked teriyaki sauce. The most incredible part? IT'S ALL UNDER $10.
Without skimping on any of the quality or taste...how?! Witchcraft. Warning -
characteristic of it's McDougal St location, Might Bowl is very very tiny. Don't expect to
find a seat, and it's a bit of a struggle to find space to wait for your order! The staff
is great though - patient and helpful. So so excited to go to school nearby.I decided to
try this place for dinner a few weeks back, because the menu looked interesting -- bowls
named after big cities in Asia. First off, the location is pretty small. Only space for 10
people to comfortably sit. I was here with 5 other people, so we took up quite a bit of
space. From the outside, it doesn't quite fit in with the bars and clubs around the area.
Anyway, I, being a big fan of Japanese food, tried the Tokyo bowl. Terrible. It has no
authentic Asian flavor whatsoever. The meat was pretty dry as well. However, my friends'
bowls all looked much better than mine. Instead of choosing a specific bowl from the menu,
they got the Make-Your-Own bowls, where you choose your rice and toppings -- similar to
Chipotle. The prices are around Chipotle price as well. After eating, I did not feel
satisfied. I wanted to go back in time and choose something else. Not sure if I'll be back
```

Figure 4. Review file after cleaning and this is important for extracting important and specific concepts.

```
0 im love look reason tri move especi come lunch mighti bowl except reason keep move everi
notasperfect place worth found one true asianfus bowl first foray menu skeptic level high
order california chicken protein avocado sprout kale mushroom top japanes mayo miso bbq
perfectli poach egg atop bed fresh ingredi left tast bud want despit overstuf stomach end
meal cours soon back anoth tri tokyo next chicken pick shitak veggi smoke teriyaki sauc
incred part 10 without skimp qualiti tastehow witchcraft warn characterist mcdougal st
locat might bowl tini dont expect find seat bit struggl find space wait order staff great
though patient help excit go school nearbi decid tri place dinner week back menu look
interest bowl name big citi asia first locat pretti small space 10 peopl comfort sit 5
peopl took quit bit space outsid doesnt quit fit bar club around area anyway big fan
japanes food tri tokyo bowl terribl authent asian flavor whatsoev meat pretti dri well
howev friend bowl look much better mine instead choos specif bowl menu got makeyourown
bowl choos rice top similar chipotl price around chipotl price well eat feel satisfi want
go back time choos someth els sure ill back anytim soon definit wont get tokyo bowl final
stop place week mayb month walk past great busi street mani restaur quick eateri refresh
find healthi asian bowl either predetermin theme bowl ie japanes inspir korean inspir etc
build born rais hawaii chose special poke bowl sushi grade ahi bed riceyour choic white
brown bonito flake scallion choic sauc chose gingersoy mayo wasabi tasti bold flavor help
sauc great textur fill 5 star space pretti tini sometim hard figur who line order who wait
also hawaiin im sure like creami mayo anyth poke soyging greatmayo idk said ate tldr easi
freshli made asian inspir bowl worth stop money wait addict mighti bowl rave one newer
creation poke bowl soooo deliciousmi hubbi love much 3 day row great dinner si week back
love manila bowl good took risk went brown rice healthier knew brown rice could tast great
```

Figure 5. A part of Sklearn output of tf-idf and there are too many “0.0”s due to unused words in the English corpus from sklearn package.

```
[0,abomin],0.0
[0,abound],0.0
[0,above20],0.0
[0,aboveaverag],0.0
[0,abovegood],0.0
[0,aboveit],0.0
[0,abras],0.0
[0,abroad],0.0
[0,abruptli],0.0
[0,absenc],0.0
[0,absent],0.0
[0,absinth],0.0
[0,absolut],0.0
[0,absolutli],0.0
[0,absorb],0.0269835452502
[0,absorpt],0.0322510086122
[0,absoult],0.0
[0,abstain],0.0
[0,abumpin],0.0
[0,abund],0.0
[0,abundantli],0.0
[0,abut],0.0
[0,abysm],0.0
[0,ac],0.0
[0,acai],0.0
[0,acanto],0.0
[0,acapulco],0.0
[0,accent],0.0
```

Figure 6. A part of MapReduce output of tf-idf. There is no “0.0”s because of self-generated corpus and save a lot space in storing the tf-idf data.

```
[0,"2pm"]       0.0278208113
[0,"3"] 0.0024387557
[0,"4"] 0.0032236306
[0,"5"] 0.0102900103
[0,"8"] 0.0122684671
[0,"975"]       0.0415051207
[0,"995"]       0.0729253762
[0,"absorb"]    0.0364626881
[0,"absorpt"]   0.0455075247
[0,"accommod"]  0.0096003927
[0,"ad"]        0.0181881386
[0,"addict"]    0.0213987686
[0,"addit"]     0.0167603634
[0,"adequ"]     0.026299168
[0,"adobo"]     0.0773307399
[0,"afford"]    0.0141365019
[0,"ago"]       0.01055421
[0,"ahi"]       0.028679713
[0,"aid"]       0.0455075247
[0,"almost"]    0.0106831621
[0,"along"]     0.0096003927
[0,"also"]      0.0001969309
[0,"altern"]    0.0232847021
[0,"alway"]     0.0028478169
[0,"amount"]    0.0050097467
[0,"amp"]       0.01785011
[0,"anchovi"]   0.0306605621
[0,"andor"]     0.0270306983
[0,"anoth"]     0.0023760815
[0,"anyth"]     0.0054263124
```

Two-pass MapReduce algorithm

MapReduce Algorithm:

Mapper1

yield word, (restaurant_index, word_count)

Reducer1

yield restaurant_index, (word, tf*idf)

Reducer2

yield (restaurant_index, word), normalized tf-idf

<u>Figure 7.</u> A part of self-generated corpus.

```
"milano" "macampchees" "cheeseburg" "content" "neue" "knoll" "courteous" "omar" "slagel" "shepherd" "chineseth" "belong"
"amazinggggg" "v20" "maneuv" "yip" "curat" "mariant" "miniantl" "hotburn" "saltier" "longand" "hum" "pre"
"strawberryblueberri" "afloat" "dedic" "unpretenti" "repair" "flock" "butterfli" "tapassmal" "citrusi" "freestyl"
"menushould" "ownermanag" "jonathan" "sandwichwithout" "slowli" "woonsan" "evas" "ikea" "forward" "threshold" "duetto"
"nonvegetarian" "rare" "hotdogburg" "locationil" "hook" "smorgasburg" "surviv" "downhil" "clock" "impos" "scrape"
"misord" "datil" "atlant" "bartenderservermanagereveryth" "luxuri" "hurri" "pale" "uninterrupt" "pina" "score" "barkyl"
"among" "obscen" "jerom" "devito" "sale" "bathroommi" "bougi" "hourandahalf" "liangpi" "overrip" "classier" "simplist"
"6pz" "snuck" "crispyin" "meticul" "aprox" "written" "biz" "chicagoan" "cajungood" "selfi" "collegi" "b1" "niec"
"futureprob" "fabl" "dug" "sachertort" "17" "indulg" "utterli" "alimentari" "volant" "oh" "unfulfil" "wellprepar"
"expect" "motto" "seagram" "bull" "sabatino" "budden" "overload" "mesh" "glorious" "saratoga" "bethani" "coke" "stilton"
"weep" "soupit" "duper" "diversey" "proport" "udon" "belt" "colomb" "inflat" "huron" "54" "accomplish" "spacer"
"chairscouch" "sonoma" "amaizin" "mark" "hoison" "delic" "slaw" "neither" "understand" "trattoria" "saliv" "forgiv"
"jalopeno" "explod" "chez" "turnstil" "marnier" "tacoy" "bang" "racial" "buddi" "cupcak" "conceiv" "tsq" "begal" "argula"
"cloud" "1pm2pm" "55th" "blackcolor" "entourag" "vendor" "arteri" "rocket" "sweetli" "instancemi" "sunset" "usa" "payday"
"hainanes" "beerssak" "surprisingli" "meatballssoppressata" "stuf" "okbut" "mustvisit" "back" "crap" "although" "citibik"
"barbi" "travel" "pick" "wheelchair" "christma" "argentina" "sand" "impend" "jose" "serviceand" "deviat" "lyonnais"
"santa" "ciroc" "faro" "jewelgem" "mochi" "ti" "calib" "fume" "minim" "croqueta" "rw" "a1" "healthyish" "southport"
"cazuela" "hodgepodg" "wellstar" "rood" "share" "huuuug" "aggress" "greenr" "businesspeopl" "grave" "recentlyi"
"tastebud" "tastehow" "inca" "comiskey" "crunchtosoft" "fuss" "win" "ignor" "nonstop" "distilleri" "exotica" "rooftop"
"breadcrouton" "itthey" "stephen" "pizzabar" "obscur" "1130am" "shawarma" "simian" "monstrou" "jog" "maneat" "coin"
"muse" "constantli" "mold" "telephon" "he" "may" "20" "restrict" "snuk" "abomin" "radio" "receipt" "so" "bw" "gunna"
"chinatownflushingelmhurst" "cosywhich" "teriyaki" "unpleas" "outstand" "knock" "rubber" "nonsoup" "poorman" "mozz"
"reconnect" "dramat" "15ish" "breakfastbrunchwhat" "cheep" "bland" "wth" "acr" "poireaux" "essenti" "fili" "ass" "paus"
"laid" "cocktaildrink" "cluni" "circu" "juicesandwichsoup" "crush" "thumbsup" "inexpensivebut" "omai" "hangov" "grate"
"stagger" "scallop" "phoflavor" "phantom" "ingredientsflavor" "rago" "skimpi" "diarrhea" "daikon" "cobbleston"
"relationship" "milliesw" "steadi" "longtim" "icecream" "playoff" "lhopit" "grassf" "sunthur" "wellknown" "devor" "tear"
"taki" "door" "oyster" "630pm" "emot" "barbalu" "badli" "blankli" "bento" "lyric" "saumon" "econom" "adjoin"
"glutenwheat" "jiggl" "mightv" "oolong" "nautic" "flop" "jul" "much" "ummmm" "jello" "sure" "superior" "thrown" "maam"
"tech" "brightlylit" "dorayaki" "posit" "tastycuz" "heh" "daughter" "surround" "honest" "steakcut" "waitressmain" "pork"
"feta" "vanna" "slightest" "violent" "trade" "siss" "breathtak" "like" "rick" "pretend" "unto" "pera" "highlight"
"postexercis" "backtrack" "dissect" "snack" "minor" "cigar" "hostgm" "gentleman" "curli" "28" "earth" "apv" "paneer"
"miso" "vacat" "throughli" "theaterstyl" "rotelli" "log" "peacheszucchini" "clutter" "deeper" "barleysw" "underdress"
"gateaux" "nomeat" "outeatin" "pari" "5bright" "picki" "haphazard" "scene" "memorabilia" "eatscoa8d" "takout" "walls"
"chines" "viscos" "fruitymartini" "nope" "quinci" "emoji" "charm" "75" "bbbbarrr" "frighten" "surreal" "amazingli"
```

<u>Figure 8.</u> Extraction of top words (k = 10 clusters, pick highest tf-idf 15 words from each cluster) using pySpark k-means implementation. Clusters output in separate files (distributed file system).

part-00000

```
(0, [2422, 8768, 4416, 1336, 2974, 7418, 9134, 3301, 7969, 3714, 12337, 8454, 8259, 2862,
5141])
(8, [1386, 314, 4775, 3309, 9759, 10555, 9153, 8329, 5121, 8105, 12318, 10263, 6441, 4035,
11061])
```

part-00001

```
(2, [555, 7887, 4917, 8329, 350, 1256, 5427, 4690, 7485, 2091, 2667, 2691, 9613, 6559,
6419])
(4, [5340, 12999, 5121, 7431, 10774, 7270, 4694, 5649, 9726, 2667, 12298, 4286, 10362,
7457, 8354])
```

part-00002

```
(6, [1844, 6571, 376, 11857, 5583, 3216, 5262, 1026, 10297, 5946, 9599, 9980, 10624, 7701,
11134])
(1, [12834, 6418, 681, 5685, 2896, 4262, 8410, 7610, 6712, 2572, 1497, 12946, 350, 8399,
9026])
```

part-00003

```
(3, [7095, 11961, 5583, 4664, 681, 4782, 7887, 258, 3216, 3391, 8188, 4917, 7745, 4618,
2734])
(9, [2814, 4416, 2091, 12337, 583, 6695, 1354, 10689, 12031, 9085, 7586, 8454, 2328, 2422,
3015])
(5, [7431, 12031, 6958, 1568, 8129, 722, 3656, 7354, 942, 12897, 5169, 10036, 8155, 4416,
9909])
(7, [2746, 10656, 3609, 3358, 5340, 7988, 8502, 7045, 5185, 9197, 12882, 1313, 2142, 8679,
6848])
```

```
['"coffe"', '"cafe"', '"sandwich"', '"cake"', '"bike"', '"m2"', '"pdq"', '"tea"', '"view"', '"caf"',
'"cuban"', '"vegan"', '"cafecito"', '"matcha"', '"pastri"']
['"taco"', '"oyster"', '"burrito"', '"seaport"', '"shrimp"', '"fish"', '"crab"', '"bowl"', '"bar"',
'"chipotl"', '"chant"', '"sum"', '"salsa"', '"mexican"', '"torta"']
['"ramen"', '"rice"', '"roll"', '"bowl"', '"pork"', '"cheesecak"', '"lobster"', '"poke"', '"marrow"',
'"sushi"', '"cocktail"', '"crudo"', '"tuna"', '"warren"', '"owl"']
['"pizza"', '"beer"', '"bar"', '"burger"', '"crust"', '"pie"', '"bartend"', '"game"', '"donut"',
'"cocktail"', '"chicago"', '"loop"', '"lou"', '"deep"', '"mac"']
['"indian"', '"samosa"', '"paneer"', '"saag"', '"curri"', '"hyde"', '"lassi"', '"soul"', '"renov"',
'"rajun"', '"indiancajun"', '"mango"', '"butter"', '"basmati"', '"newli"']
['"dumpl"', '"tapa"', '"noodl"', '"shanghai"', '"french"', '"brunch"', '"wine"', '"soup"', '"duck"',
'"waffl"', '"breakfast"', '"egg"', '"pork"', '"pancak"', '"sake"']
['"thai"', '"pho"', '"curri"', '"pad"', '"noodl"', '"roti"', '"rice"', '"teriyaki"', '"hyde"',
'"korean"', '"jerk"', '"roll"', '"fidi"', '"asian"', '"cash"']
['"smoothi"', '"sandwich"', '"sushi"', '"cuban"', '"bagel"', '"avocado"', '"zeu"', '"potbelli"',
'"gyro"', '"margon"', '"deli"', '"vegan"', '"crepe"', '"coffe"', '"church"']
['"burger"', '"gyro"', '"dog"', '"dmk"', '"mikkey"', '"fri"', '"breakroom"', '"frite"', '"patti"',
'"branko"', '"medici"', '"xian"', '"curd"', '"sandwich"', '"uic"']
['"pasta"', '"italian"', '"spaghetti"', '"ravioli"', '"pizza"', '"linguin"', '"lasagna"', '"itali"',
'"tiramisu"', '"calamari"', '"gnocchi"', '"clam"', '"bread"', '"bolognes"', '"ink"']
```

Figure 9. Extraction of top words (k = 10, pick highest tf-idf 15 words from each cluster) using sklearn k-means (for comparison with spark implementation).

```
["brunch" "waffl" "french" "toast" "view" "egg" "breakfast" "benedict" "jane" "duck" "avocado"
"burger" "le" "croqu" "pancak" ]
["tapa" "spanish" "sangria" "brava" "wine" "jamon" "spain" "marrow" "patata" "octopu" "charcuteri"
"chorizo" "suprema" "bocadillo" "steak" ]
["pasta" "italian" "spaghetti" "ravioli" "linguin" "itali" "clam" "wine" "tiramisu" "gnocchi" "panini"
"calamari" "dessert" "meatbal" "bread" ]
["taco" "burger" "bar" "beer" "cocktail" "bartend" "oyster" "seaport" "tot" "lobster" "drink"
"sandwich" "mac" "game" "cheesecak" ]
["pizza" "crust" "lou" "pasta" "deep" "italian" "lasagna" "pie" "pepperoni" "medici" "robert"
"giordano" "olio" "andiamo" "malnati" ]
["jerk" "smoothi" "hyde" "vegan" "donut" "healthi" "mikkey" "sandwich" "breakfast" "bike" "wrap"
"park" "lyfe" "plantain" "nando" ]
["burger" "gyro" "sandwich" "sub" "uic" "chant" "dog" "dmk" "m2" "branko" "coffe" "zeu" "student"
"cafe" "italian" ]
["dumpl" "shanghai" "noodl" "soup" "rice" "chinatown" "xlb" "sum" "chines" "pork" "cheung" "dim"
"shanghaines" "roll" "clay" ]
["noodl" "thai" "curri" "pad" "roti" "pho" "rice" "indian" "dumpl" "xian" "malaysian" "soup" "boiler"
"cajun" "samosa" ]
["bowl" "sushi" "ramen" "rice" "poke" "teriyaki" "japanes" "chipotl" "roll" "pho" "bbq" "pork"
"burrito" "miso" "tuna" ]
```

Figure 10. Univariate feature selection with Chi-square metric.

```
b = SelectKBest(chi2, k=40)
X_new = b.fit_transform(X_train, y_train)
print X_train.columns[b.get_support()]
```

```
Index([u'brunch', u'shanghai', u'pad', u'rice', u'vegan', u'game', u'ramen',
       u'cheesecak', u'dumpl', u'cafecito', u'fish', u'burger', u'lobster',
       u'poke', u'sum', u'smoothi', u'pasta', u'curri', u'owl', u'duck',
       u'beer', u'crab', u'lou', u'm2', u'frite', u'thai', u'donut', u'hyde',
       u'breakroom', u'deep', u'branko', u'bowl', u'torta', u'roll', u'pho',
       u'salsa', u'tuna', u'oyster', u'burrito', u'wine'],
      dtype='object')
```

Figure 11. Alternative feature selection method, random forest feature importance.

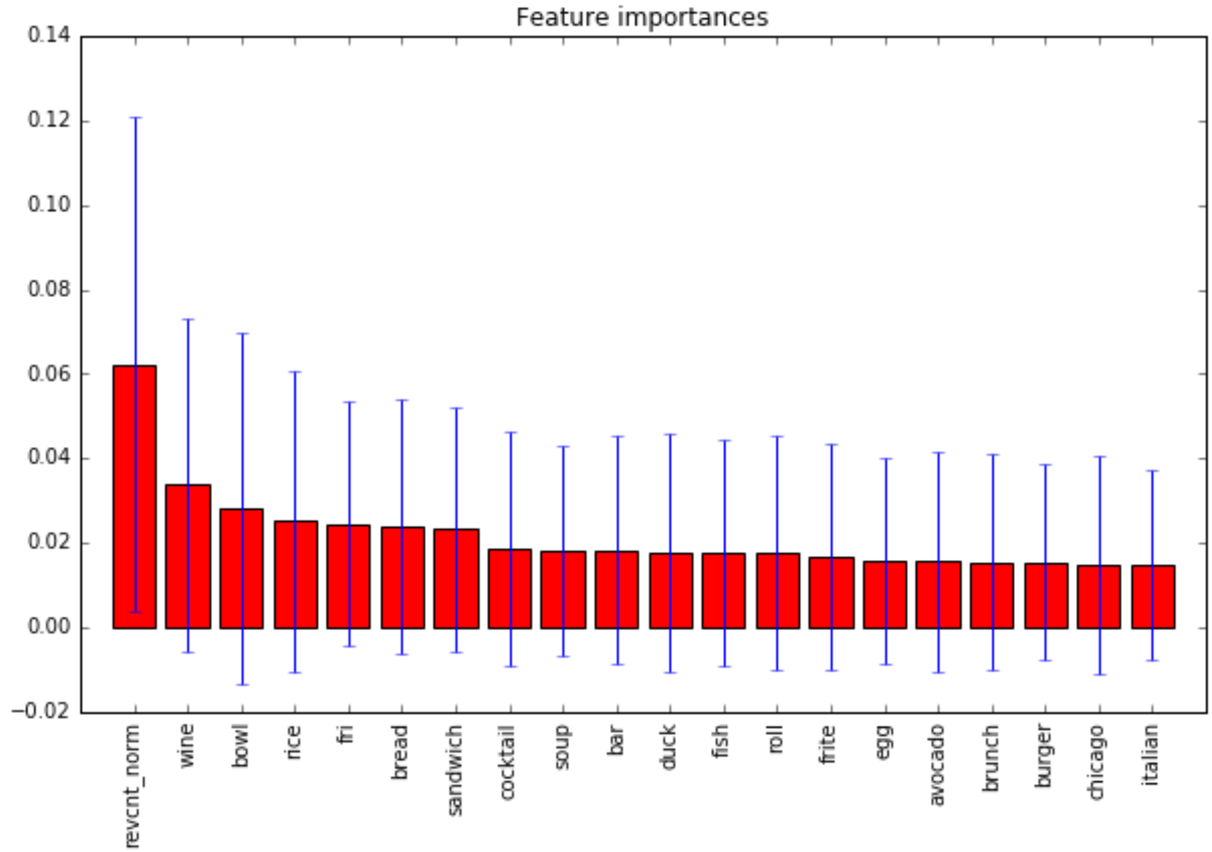
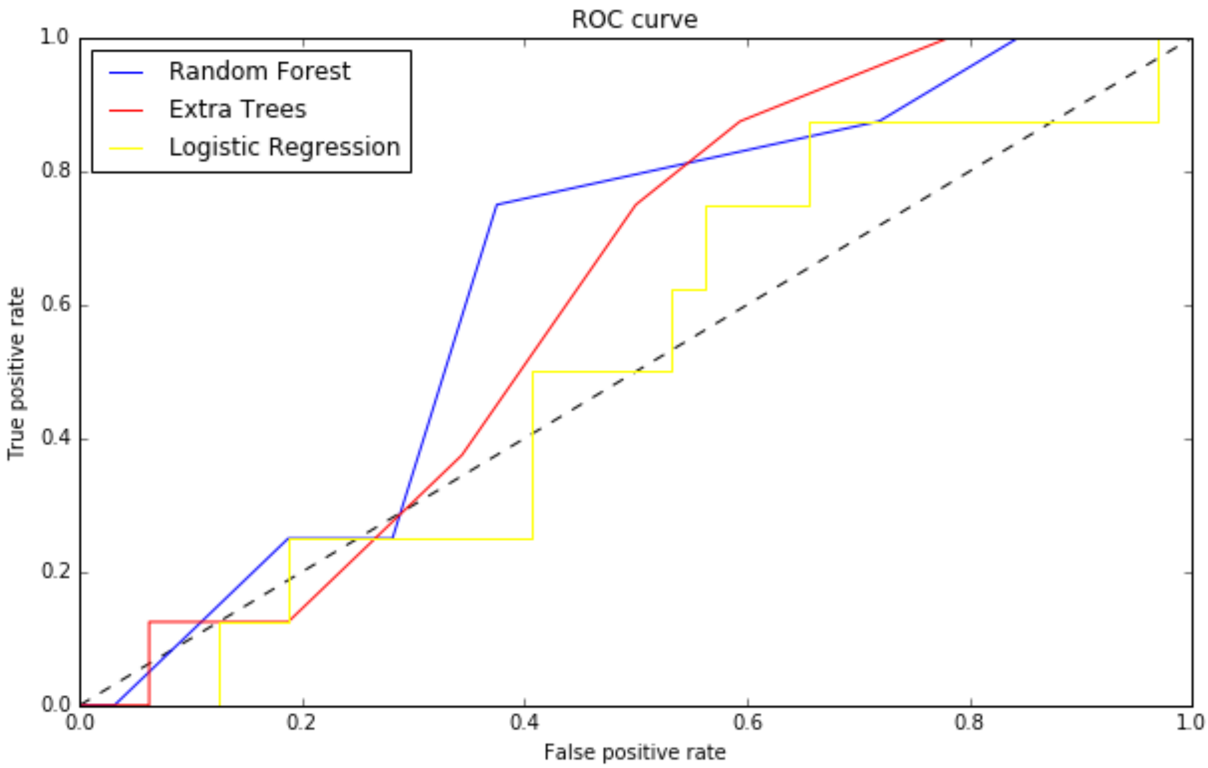


Figure 12. Comparison of models by cross-validation performance (70% split of dataset). No significant different is observed within the three models.

Model	Parameters	F1_weighted score
Random Forest Classifier	n_estimators: 10 max_depth: 100 “1” class weight: 5	0.66
Extra Trees Classifier	n_estimators: 10 max_depth: 90 “1” class weight: 3	0.66
Logistic Regression	C: 0.1 “1” class weight: 3	0.65

Figure13. ROC for model performance on holdout verification dataset (30% split of dataset). Random Forest performs better than Extra trees and logistic regression model is meaningless. Performance on the verification set is much worse than it in the cross-validation so over-fitting problems need to be addressed to improve the model performance.



Ways to improve model performance:

There is a lot of room for the model improvement and the following are some aspects that could by tried in the future.

1) Get more data (restaurants) from more diverse locations.

2) Optimize k for k-means.

3) Try various clustering techniques.

4) Process the reviews in more details such as segmentats in verb, abjective, noun or investigate the positive and negative sentiment.

5) Set binary representation of term rather than the count to avoid the bias in regards of the different count of reviews for each restaurant.

6) Test on feature selections more precisely.

7) Test with more classification models (e.g. SVM, Naïve Bayes).

8) Use more techniques to deal with unbalanced class labels (e.g. undersampling, oversampling).

Project Responsibility:

- Qian Wang:

  1) Review retrieval and test on small size of request; 2) Review clean; 3) MapReduce implementation of tf-idf. 4) Spark implementation of k-means; 5) Classification models.

- Qianyu Deng:

  1) Test on a potential limit on web crawler and generate big dateset with reviews; 2) Sklearn implementation of tf-idf transformation; 3) Sklearn implementation of k-means.