Yelp Restaurant Review Clustering System

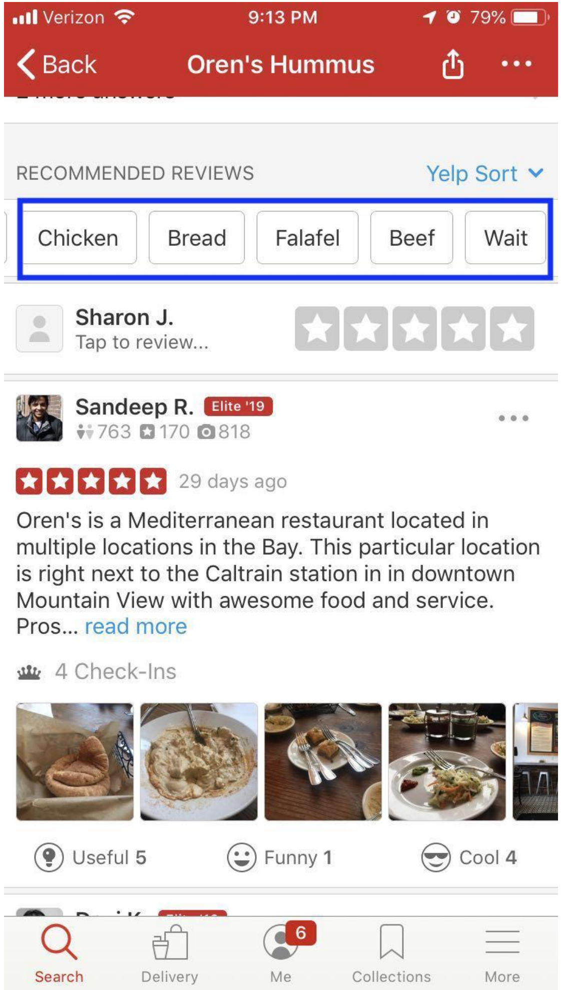
Team 9
Siyun Liang Chunwei Lee Xiaoting Jin

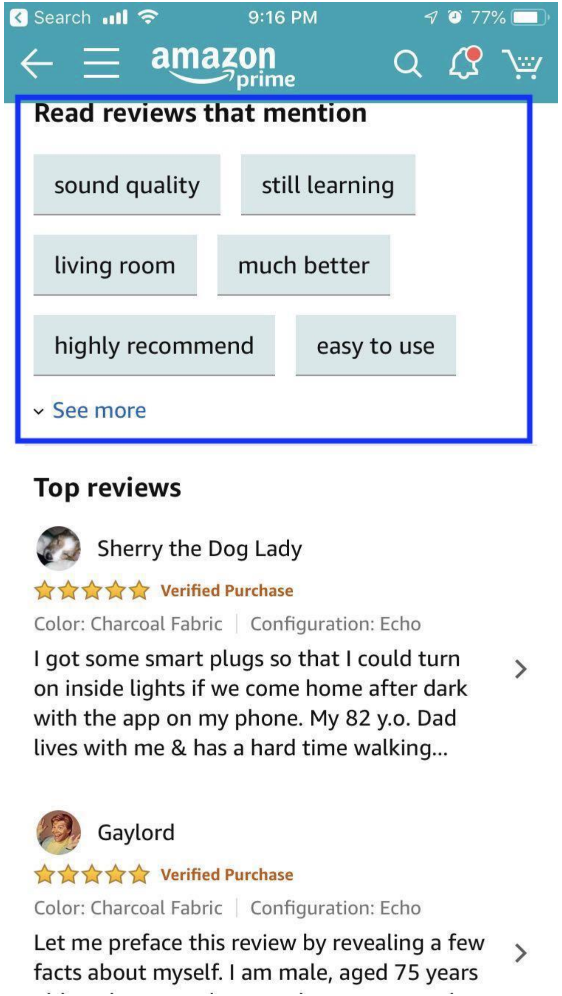
Project Goal

Can we add a feature like "Read reviews that mention" in Yelp mobile APP?

Our Solution

Clustering and more...





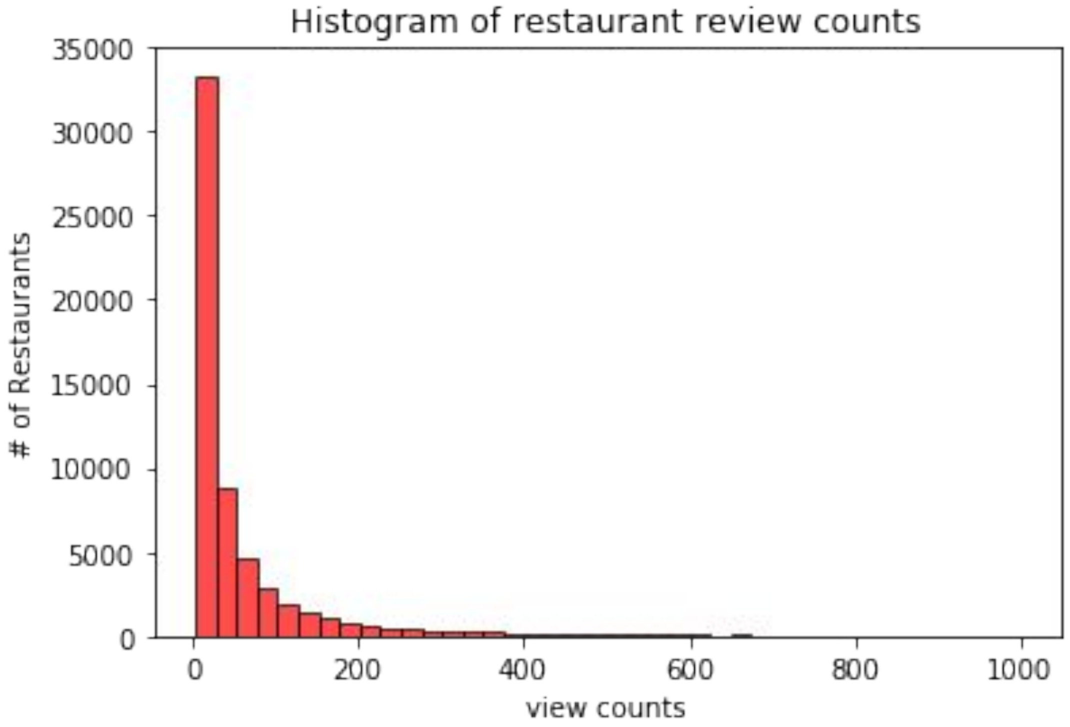
Dataset

Yelp dataset

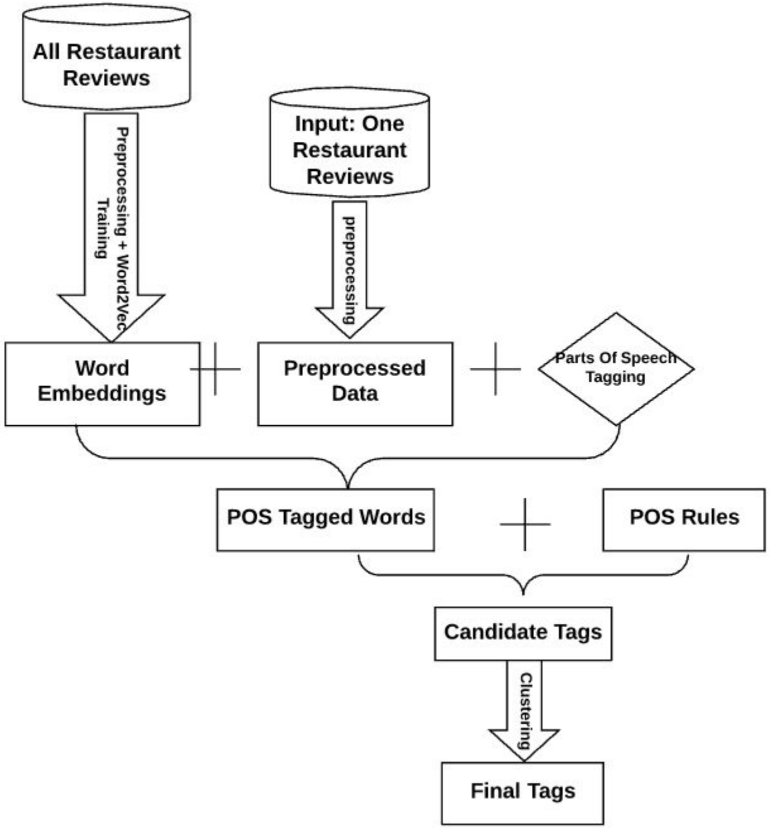
business.json
- 138.3 MB
- 192,609 samples
- 27 feature

review.json
- 4.71 GB
- 6,685,900 samples
- 9 feature

Filter Restaurant business
- 59,371 restaurants
- 4,201,684 reviews
- Avg: 71 reviews / business
- Only review ‘text’ feature is used



Approach 1 Word Embedding



1 **word embeddings after preprocess**

Representative Vectors of Single Word

```
model.wv.get_vector("terrible")
```

```
array([ 1.03065252e+00, -5.27238011e-01, -1.73737311e+00, -1.96166575e+00,
        6.54447198e-01, -1.71924103e-02, -2.79911637e-01, -1.73708844e+00,
       -9.17404532e-01, -2.74804735e+00, -2.04840317e-01, -1.36049771e+00,
        2.61394501e-01, -2.60879636e-01, -9.00094509e-01, -7.59506896e-02,
       -1.00086391e+00, -2.06596375e+00,  3.30865175e-01, -2.31672955e+00,
        4.99523729e-01, -1.47431993e+00,  5.54392338e-01,  2.49320817e+00,
        3.91174221e+00,  8.03913832e-01,  1.78700888e+00, -4.20244455e-01,
        2.08787894e+00,  5.95218241e-01,  9.89763975e-01, -1.76790357e+00,
```

```
: model.wv.most_similar('terrible')
```

```
: [('horrible', 0.9798260927200317),
   ('awful', 0.9358709454536438),
   ('horrendous', 0.8255303502082825),
   ('lousy', 0.8194860219955444),
   ('horrid', 0.8093457221984863),
   ('bad', 0.806841254234314),
   ('poor', 0.7922376394271851),
   ('atrocious', 0.7796613574028015),
   ('horrific', 0.7734644412994385),
   ('subpar', 0.7334288358688354)]
```

Approach 1 Word Embedding

2 Extract Candidate Tags

Part-Of-Speech Tagger (**POS** Tagger)

Rules	Examples
adj + noun or noun + adj	fantastic menu, poor quality, dirtiest places, prices reasonable
adv + verb or verb + adv	highly recommend, sat immediately
auxiliary + verb	must try
noun + noun	garlic bread, bottle wine
adj + to + verb	easy to use
…….	

2 Tags Clustering

DBSCAN: 1. Results highly depend on epsilon and minPts
2. Output number of tags unstable

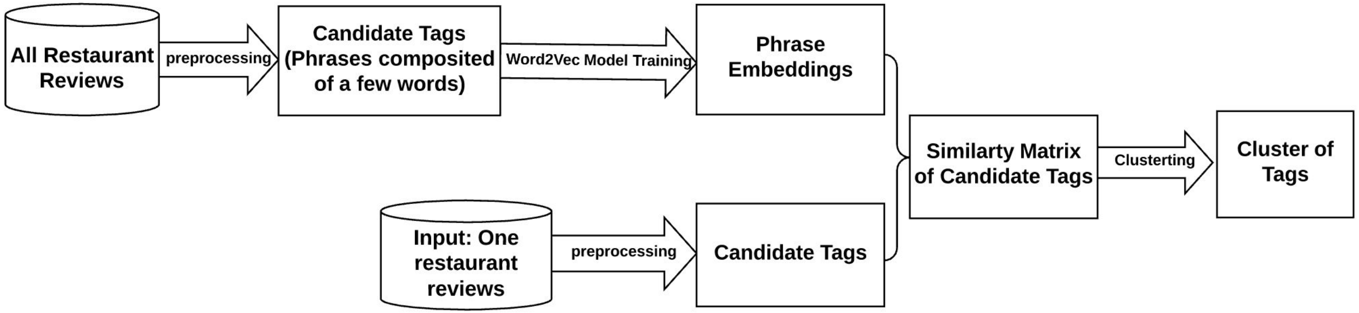
```
prediction(reviews,0.25,8)[0]
```

```
[['really', 'good'],
 ['going', 'back'],
 ['fresh', 'ingredients'],
 ['best', 'lunchdinner'],
 ['food', 'delicious'],
 ['especially', 'turkey', 'burger'],
 ['fast', 'busy']]
```

```
prediction(reviews,0.25,5)[0]
```

```
[['really', 'good'],
 ['going', 'back'],
 ['fresh', 'food'],
 ['healthy', 'options'],
 ['best', 'lunchdinner'],
 ['food', 'delicious'],
 ['reasonable', 'prices'],
 ['usual', 'selection'],
 ['turkey', 'burger'],
 ['sweet', 'potato'],
 ['service', 'impersonal'],
 ['tables', 'available'],
 ['cheese', 'drives'],
 ['red', 'salad'],
 ['forgot', 'add', 'order'],
 ['disappointing', 'experience
 ['place', 'lunch']
```

Approach 2 Phrase Embedding



1 Using stop word to generate candidate phrases

2 Using word2vec to compute similarity

```
model.wv.most_similar('great_place')
```

```
[('goodgreat_place', 0.5212652087211609),
 ('excellent_spot', 0.4757899045944214),
 ('placegreat_place', 0.46101465821266174),
 ('perfect_place', 0.45851147174835205),
 ('ideal_place', 0.457986056804657),
 ('friendlygreat_place', 0.451860249042511),
 ('excellentgreat_place', 0.45150217413902228),
 ('perfectgreat_place', 0.4467985928058624),
 ('deliciousgreat_place', 0.43668505549430847),
 ('nicegreat_place', 0.43252062797546387)]
```

3 Clustering to generate tags w.r.t restaurant

DBSCAN

Epsilon and min_sample is set according to number of reviews
Try to use Silhouette Coefficient to tune

K-Medoid

Tags are generated from the medoid points, therefore the results might be different for different initial points.

Affinity Propagation

No need to set cluster number for input parameter

```
print(get_predict('ikCg8xy5JIg_NGPx-MSIDA')) ##review number=15
```

```
Silhouette Coefficient: 0.175
['dirtiest_places', 'health_inspections', 'rude_staff', 'kensington_location']
```

```
print(get_predict('zvO-PJCpNk4fgAVUnExYAA'))  ##review number=29
```

```
Silhouette Coefficient: 0.195
['sports_grill', 'sports_bars', 'pool_tables', 'playoff_games']
```

```
print(get_predict('ERnG-1q3igX3VSgm5uLZ6A')) ##review number=74
```

```
Silhouette Coefficient: -0.092
['pizza_pazzi', 'favourite_restaurant', 'tomato_sauce', 'favourite_dish', 'order_pasta', 'great_service', 'balsamic_
vinaigrette', 'second_time', 'reasonably_priced', 'rice_balls']
```

```
print(get_predict("eU_713ec6fTGNO4BegRaww")) ##review number=135
```

```
Silhouette Coefficient: -0.102
['crab_tortellini', 'highly_recommend', 'best_italian_food', 'large_party', 'garlic_bread']
```

```
print(get_predict('KR2kRmHnRCaNzOUEGoB25w')) ##review number=279
```

```
Silhouette Coefficient: -0.119
['lola_fries', 'crocker_park', 'dont_know', 'pulled_pork', 'onion_rings', 'happy_hour', 'seated_right_away', 'rosema
ry_fries', 'veggie_burgers']
```

```
print(get_predict('Bf2fuqWbHd3L-X69FSMvmg'))##review number=313
```

```
Silhouette Coefficient: 0.107
['kensington_market', 'mexican_food', 'fish_tacos', 'pico_gallo', 'fish_taco', 'chicken_tacos']
```

Evaluation

- How is the tags' variety?

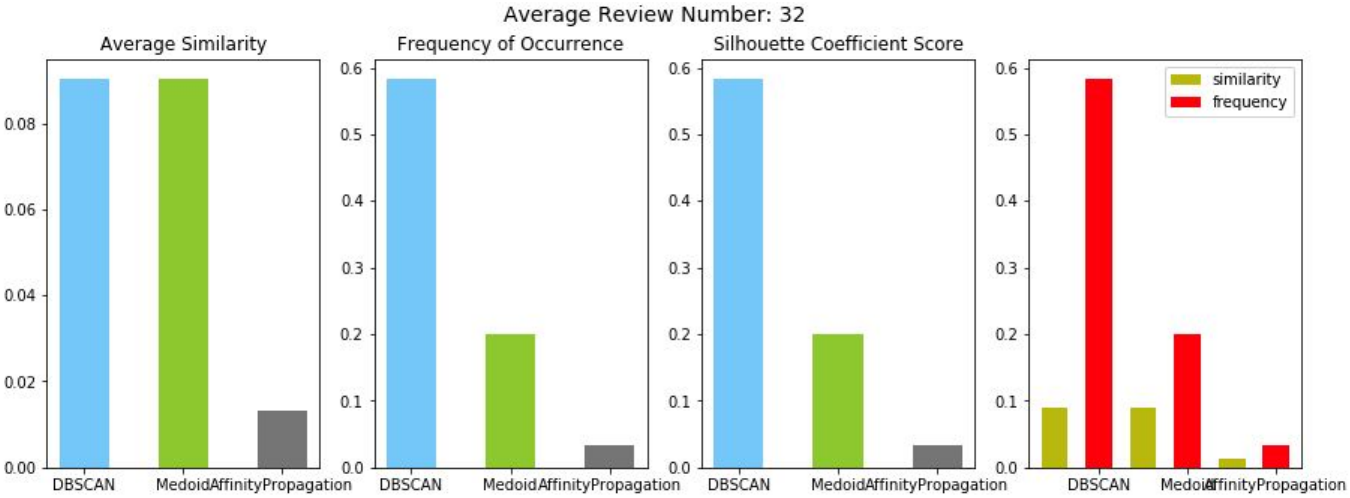
  Average Similarity between Tags

- Are they often mentioned?

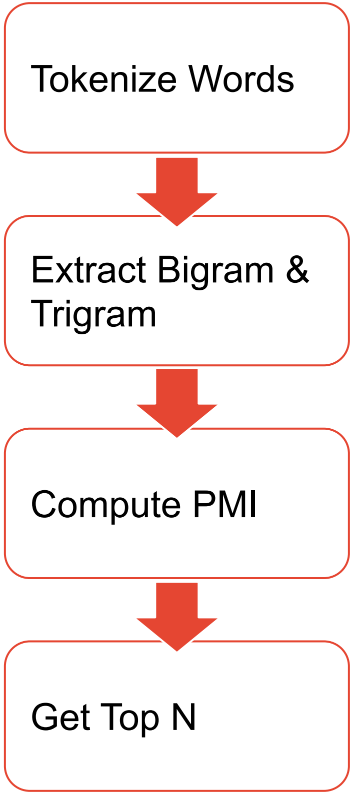
  Frequency of Tag Occurrence

- Evaluate Clustering

  Silhouette score



Approach 3 By PMI




```
quickversion('ikCg8xy5JIg_NGPx-MSIDA')##redelicview number=15
```

```
['lunch_steak_sandwich', 'past_health_inspection']
```

```
quickversion('zvO-PJCpNk4fgAVUnExYAA') ##review number=29
```

```
['sweet_potato_fry', 'worth_spending_money', 'happy_hour', 'pool_table', 'service_slow', 'watch_game', 'playoff_game
']
```

```
quickversion('ERnG-1q3igX3VSgm5uLZ6A')  ##review number=74
```

```
['fried_risotto_ball', 'authentic_neapolitan_pizza', 'tomato_sauce', 'authentic_italian', 'italian_food', 'pizza_paz
zi']
```

```
quickversion("eU_713ec6fTGNO4BegRaww")  ##review number=135
```

```
['pepper_cream_sauce', 'shrimp_crab_tortellini', 'italian_wedding_soup', 'best_italian_food', 'cooking_class', 'high
ly_recommend', 'garlic_bread', 'tavola_italiana', 'large_group']
```

```
quickversion('Bf2fuqWbHd3L-X69FSMvmg')##review number=313
```

```
['authentic_mexican_food', 'pico_gallo', 'kensington_market', 'corn_tortilla', 'fish_taco', 'mexican_restaurant', 't
aco_pastor']
```

```
quickversion('NyLYY8q1-H3hfsTwuwLPCg') ##review number=547
```

```
['hand_washing_station', 'chicken_tikka_masala', 'tikka_masala_bowl', 'tikka_masala_sauce', 'chipotle_indian_food',
'indian_fast_food', 'highly_recommend', 'lamb_meatball', 'mango_lassi', 'samosa_chaat', 'fast_casual', 'wheat_naan']
```

```
quickversion('8mIrX_LrOnAqWsB5JrOojQ') ##review number=1289
```

```
['super_mario_bros', 'pinball_hall_fame', 'row_row_pinball', 'school_arcade_game', 'classic_arcade_game', 'row_pinba
ll_machine', 'salvation_army', 'donkey_kong', 'highly_recommend', 'memory_lane', 'star_war', 'star_trek', 'go_charit
y']
```

Discussion

Lack of powerful metrics

- Difficult for model tuning and selection
- Previous studies usually employed user testing or A/B test

Representative and easy-to-understand tags

- **Word embedding approach** would miss some meaningful tags due to the difficulty of covering all the rules that are not always work for special combinations.
- **Phrase embedding approach** is most likely to produce nonsense tags.it is possible to lose meaningful data as well since it filter out the single word between stop words.
- **PMI approach** is likely to generate similar tags.