Capstone Presentation

A Hybrid Recommendation System with Yelp Challenge Data

Team Yelper_Helper

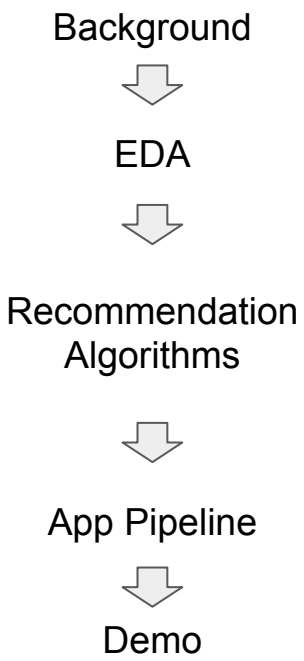
S. Kickham, S. O'Mullane, R. Rad, A. Rubino, C. Shi

NYC Data Science Academy Cohort 9

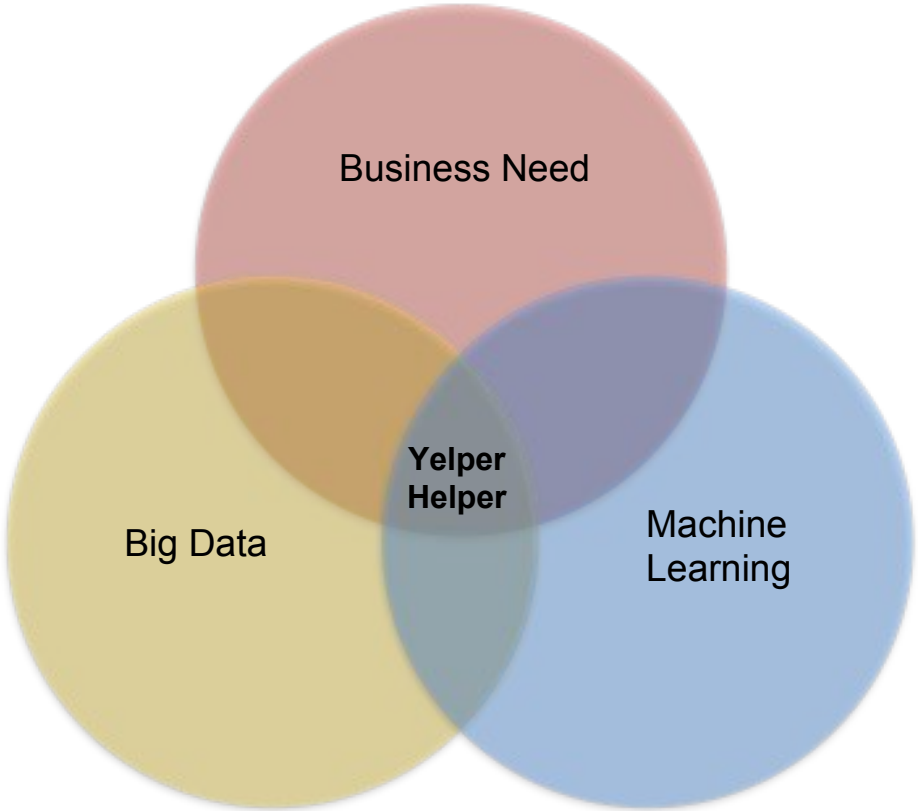
06/20/2017

Outline

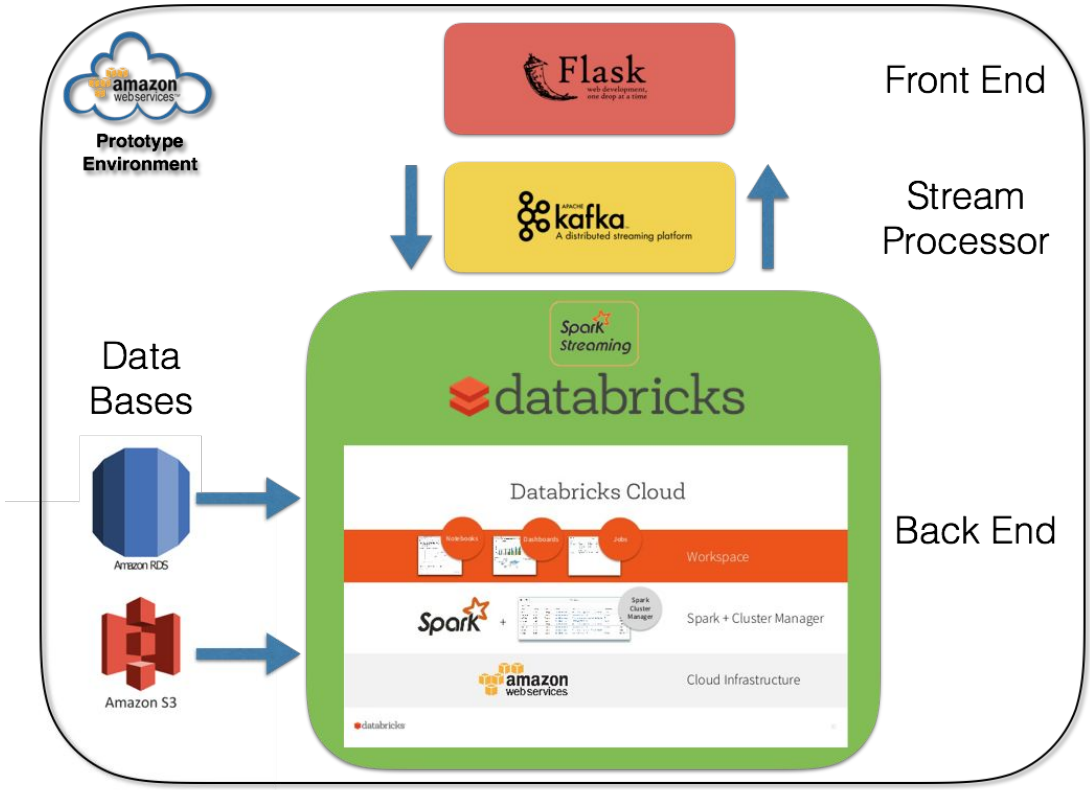
- Motivation
- Project Summary
- Data Source
- Data Preparation and Exploratory Analysis
- Hybrid Recommendation System
  - Natural Language Processing (NLP)
  - Collaborative Filtering (CF)
  - Social Network
  - Location-based
- Data Pipeline
  - Recommendation Engine
  - App Workflow (Flask - Kafka - Rec Engine - Kafka - Flask)
- Live Demo
- Lessons Learned and Future Work
- Acknowledgement



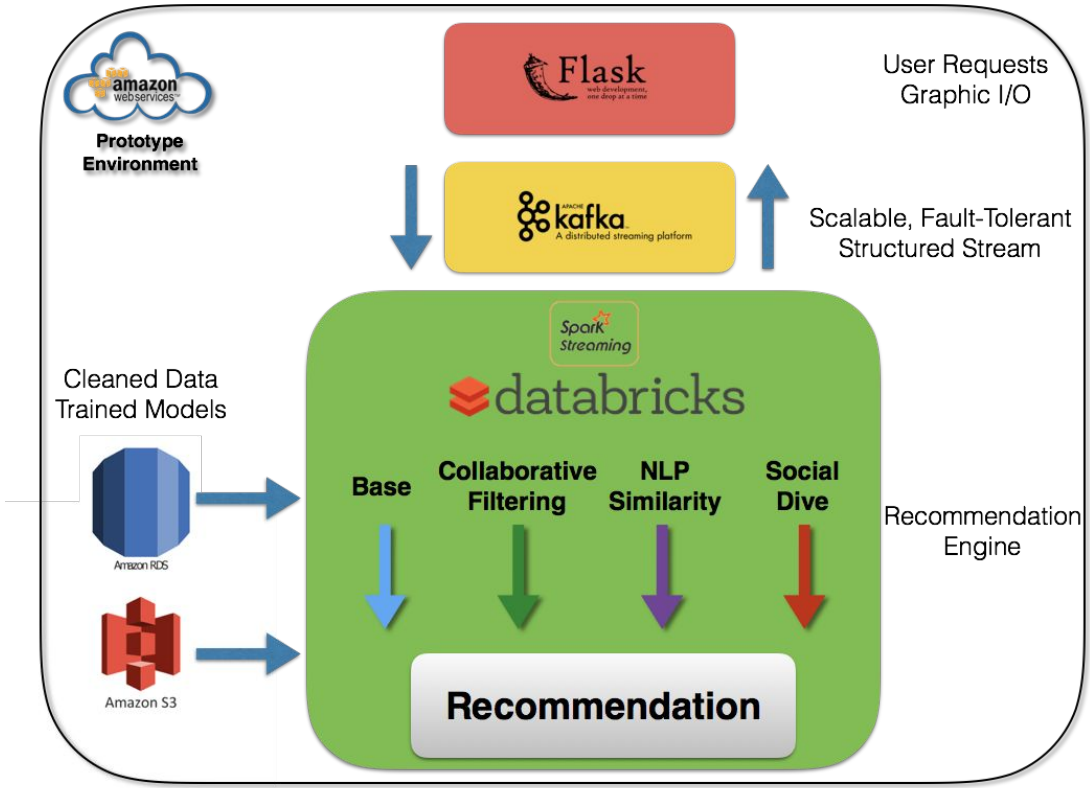
Motivation



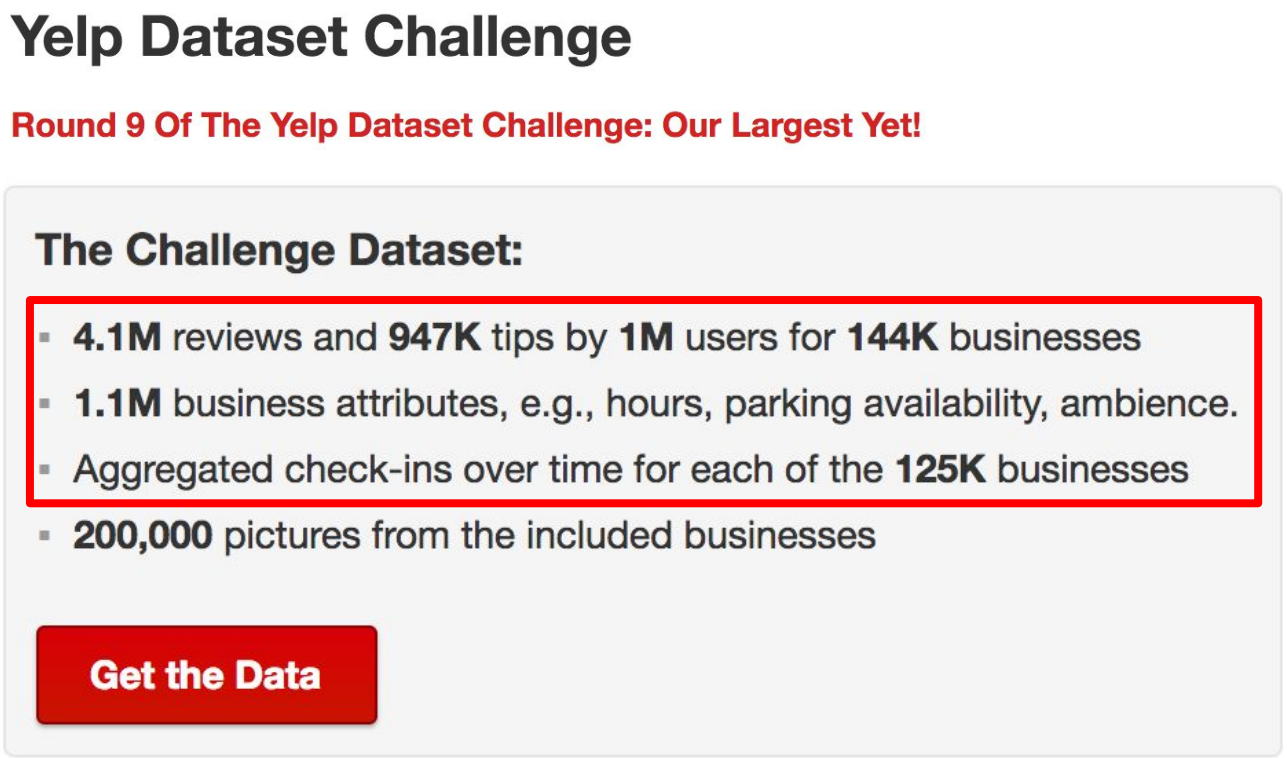
Project Architecture



Project Architecture



Data Source



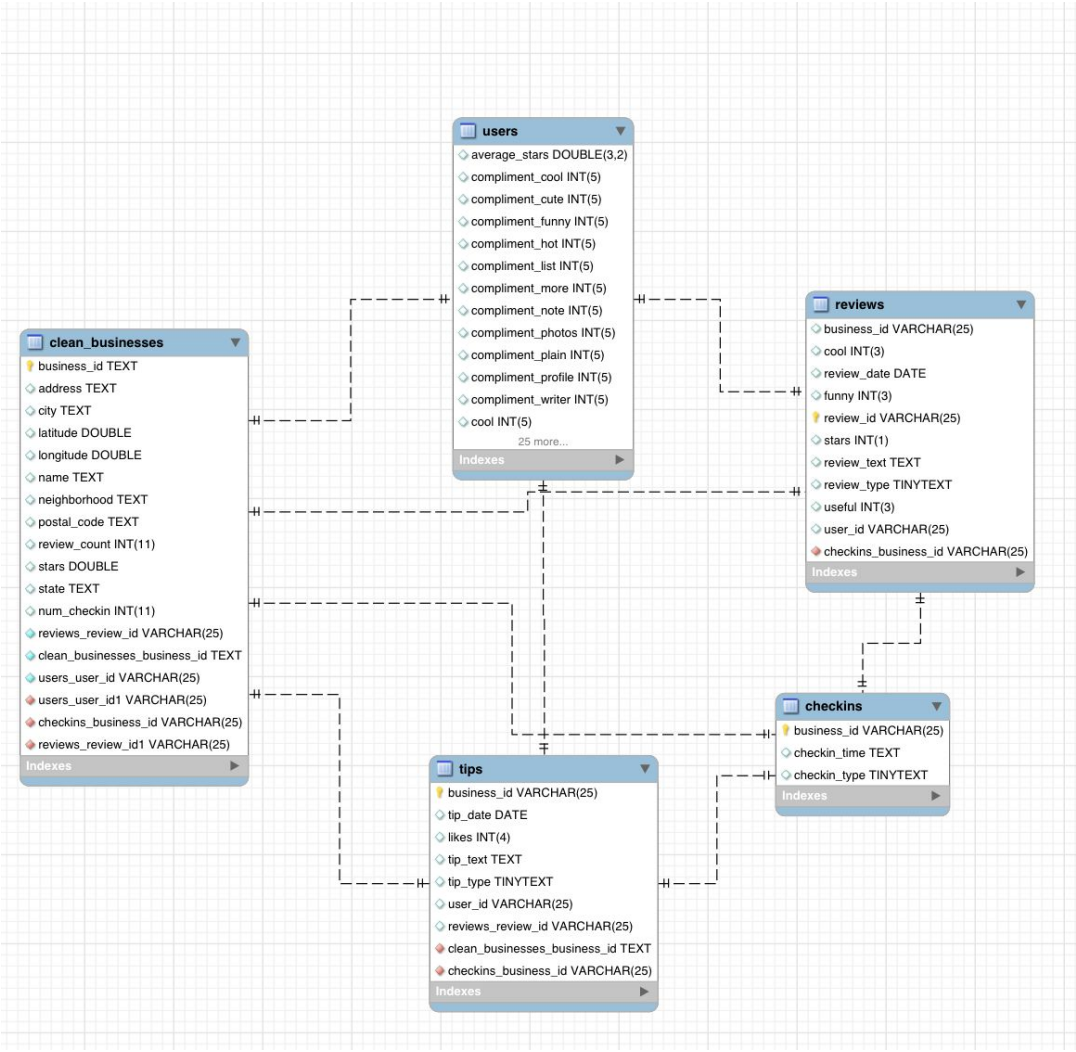
5 Gb text data
json format

Amazon RDS - MySQL Database

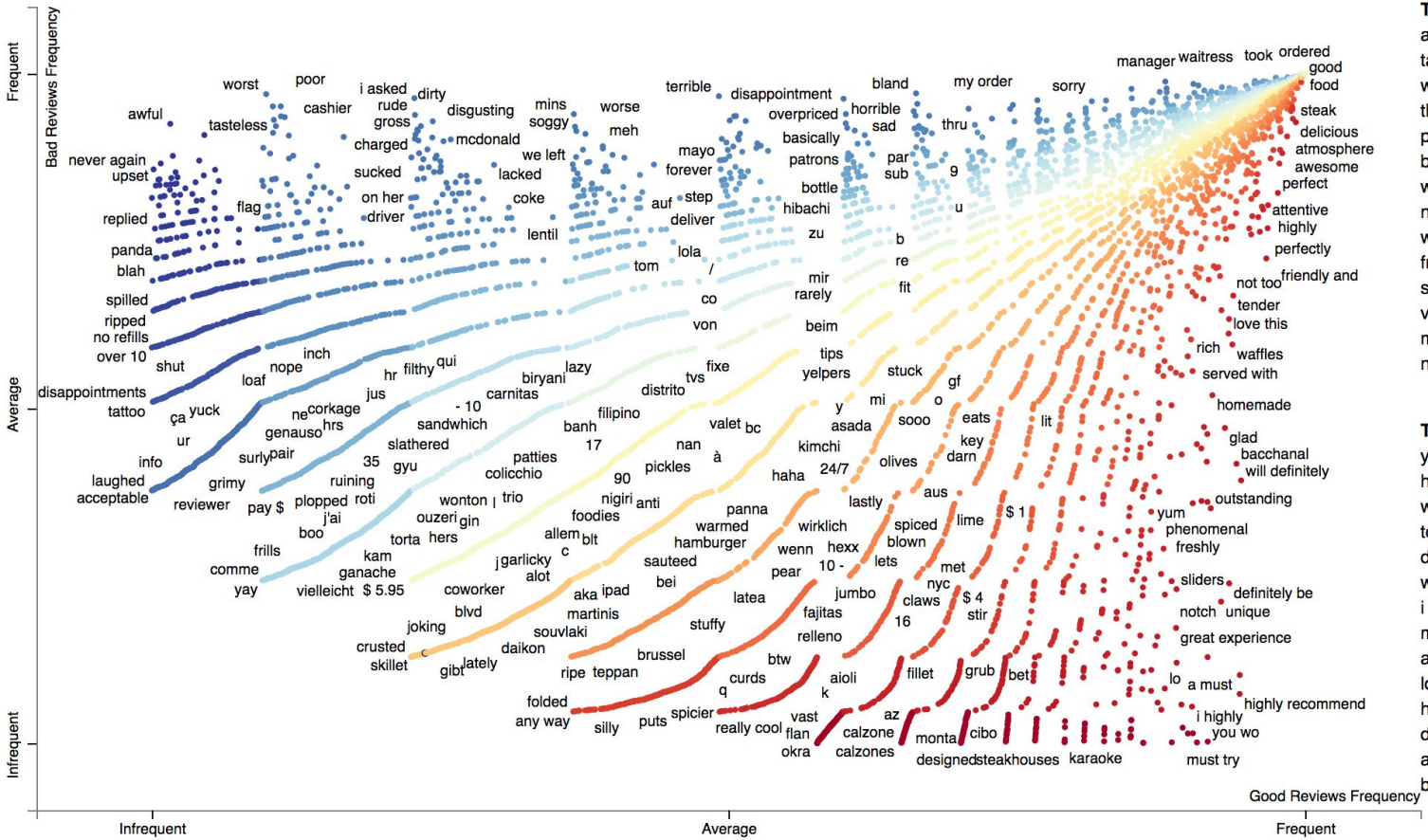
- RDS = Relational Database Service
- Main tables include:
  - **Reviews - 4.1M**
  - **Tips - 947K**
  - **Businesses - 144K**
  - **Checkins - 125K**
  - **Users - 1M**
- The main tables allow for simple subtable creation, which allows for fast requests and data loads.

MySQL Schema Viz

- Primary Keys for all tables:
  - Businesses: business_id
  - Users: user_id
  - Reviews: review_id
  - Tips: user_id
  - Checkins: business_id
- RDS allowed us to create relationships between tables.



Word Frequency per Review Type



Top Bad Reviews

awful
tasteless
worst
the worst
poor
burnt
we asked
not even
will never
frozen
sucks
very disappointed
management
never again

Top Good Reviews

you wo
highly recommend
would definitely
to die
die for
well worth
i highly
must try
a must
loved it
hands down
definitely recommend
are amazing
boba

Characteristic

ramen
bacchanal
waitress
tacos
yelp
tasted
waiter
fries
wagyu
frites
flavorful
sashimi
overcooked
ribeye
groupon
ayce
craftsteak
filet
flavorless
guac
soggy
buffets
poutine
brisket
crispy
sushi
calamari
hibachi
appetizer
overpriced

Bad Reviews document count: 1,000; word count: 143,367
Good Reviews document count: 1,000; word count: 112,809

Sentiment Analysis

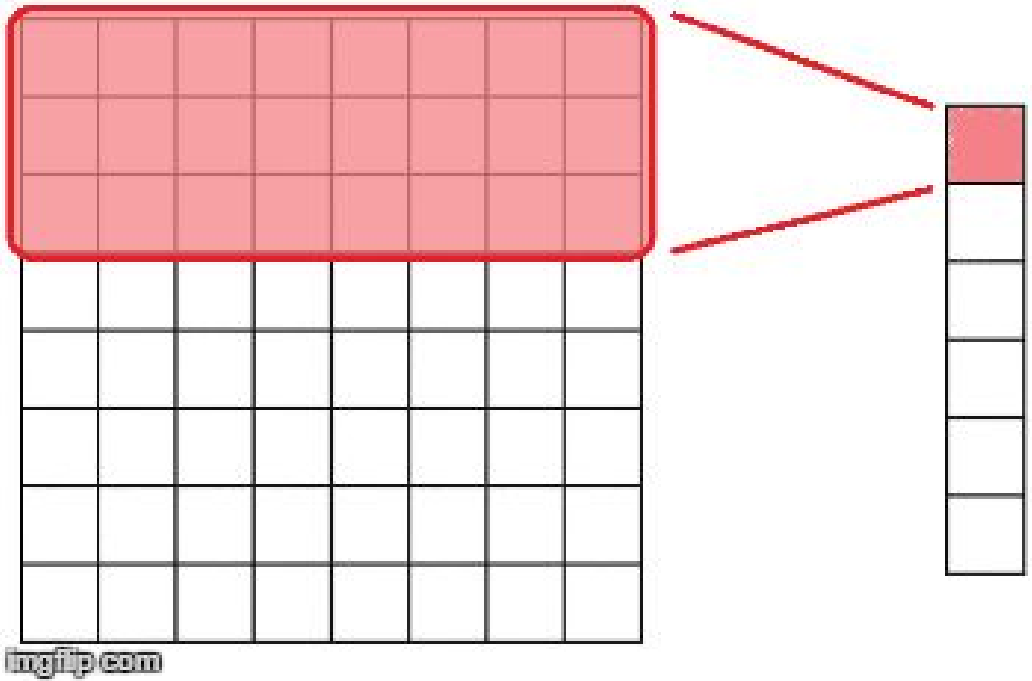
1. Binarize
2. Tokenize
3. Texts_to_Seq

1. Supervised
2. Convolution
3. Pooling

1. Results
2. Spot Check
3. Base Rec.

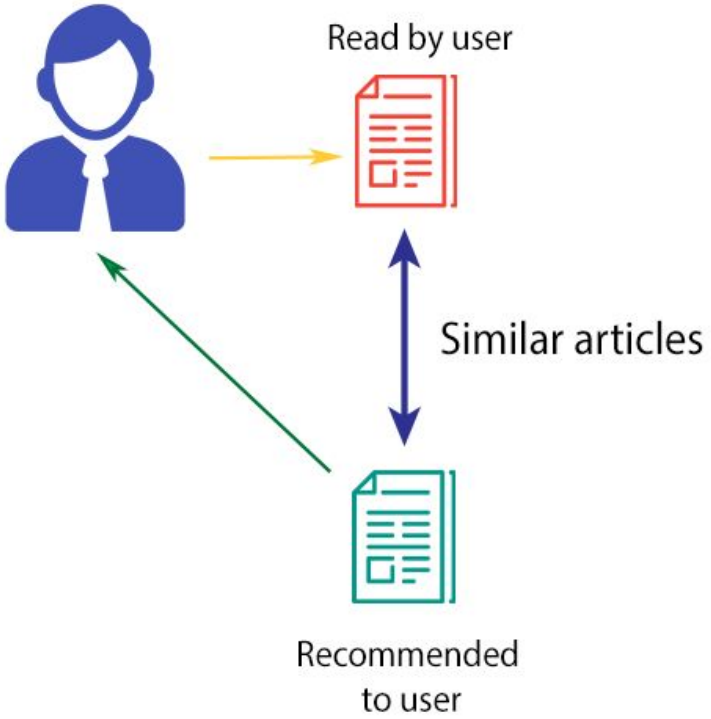
bit.ly/sent_analysis

Convolution and Pooling



Content-Based Recommendations

- Item-to-item based on user profile
- Keyword soft filtering
- Average Word2Vec



http://i65.tinypic.com/2ebah6c.png

Cosine Similarity using Word2Vec

1. Tokenize
2. Stop Word Removal

1. Unsupervised
2. Skip-gram
3. Hyperparameters

1. Cosine Similarity
2. Rank

bit.ly/word_2_vec

Word Similarity

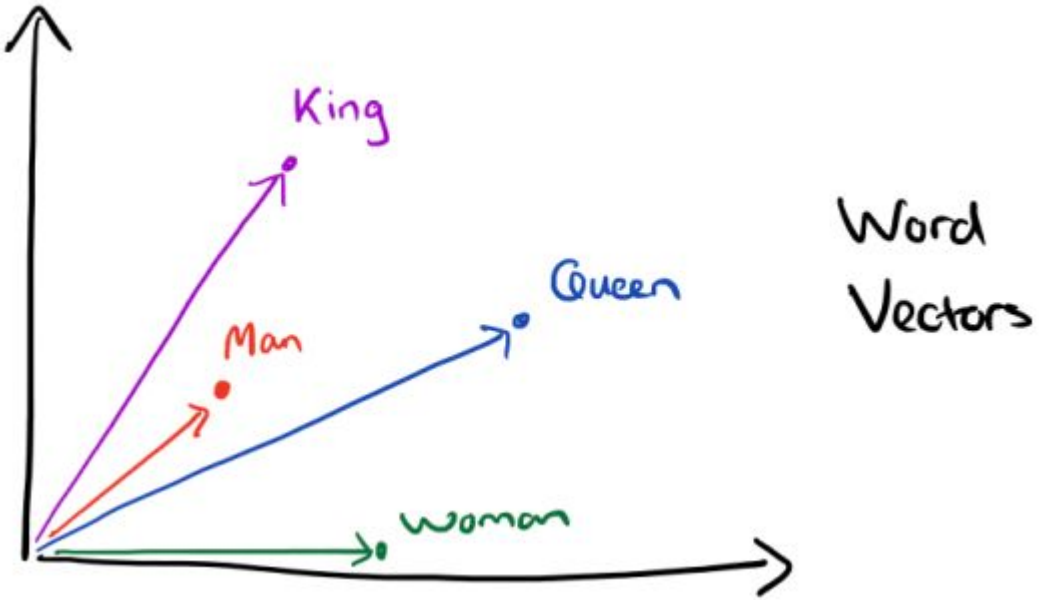
Cmd 28

```
1 #test similarity between words
2 synonyms = model.findSynonyms("chinese", 5)
3 synonyms.show(5)
```

```
+------------+------------------+
|        word|        similarity|
+------------+------------------+
|       asian|0.7530649328506537|
|   cantonese|0.7275205535026946|
|    filipino|0.7259573442238313|
|shanghainese|0.6988964525834206|
|      korean|0.6691107471966923|
+------------+------------------+
```

Command took 0.48 seconds -- by skickham@gmail.com at 6/15/2017, 2:59:40 PM on My Cluster

Word Similarity



https://adriancolyer.files.wordpress.com/2016/04/word2vec-king-queen-vectors.png?w=600

Word Algebra

https://adriancolyer.files.wordpress.com/2016/04/word2vec-king-queen-composition.png?w=600

Word Algebra: Food

https://adriancolyer.files.wordpress.com/2016/04/word2vec-king-queen-composition.png?w=600

Word Algebra: Food

Word Algebra: Examples

'filet_mignon' - 'beef' + 'seafood' =

'fork' + 'soup' =

'drink' + 'barley' =

```
In [*]: word_algebra(add=[u'filet_mignon', u'seafood'], subtract=[u'beef'])
        word_algebra(add=[u'fork', u'soup'])
        word_algebra(add=['drink', 'barley'])
```

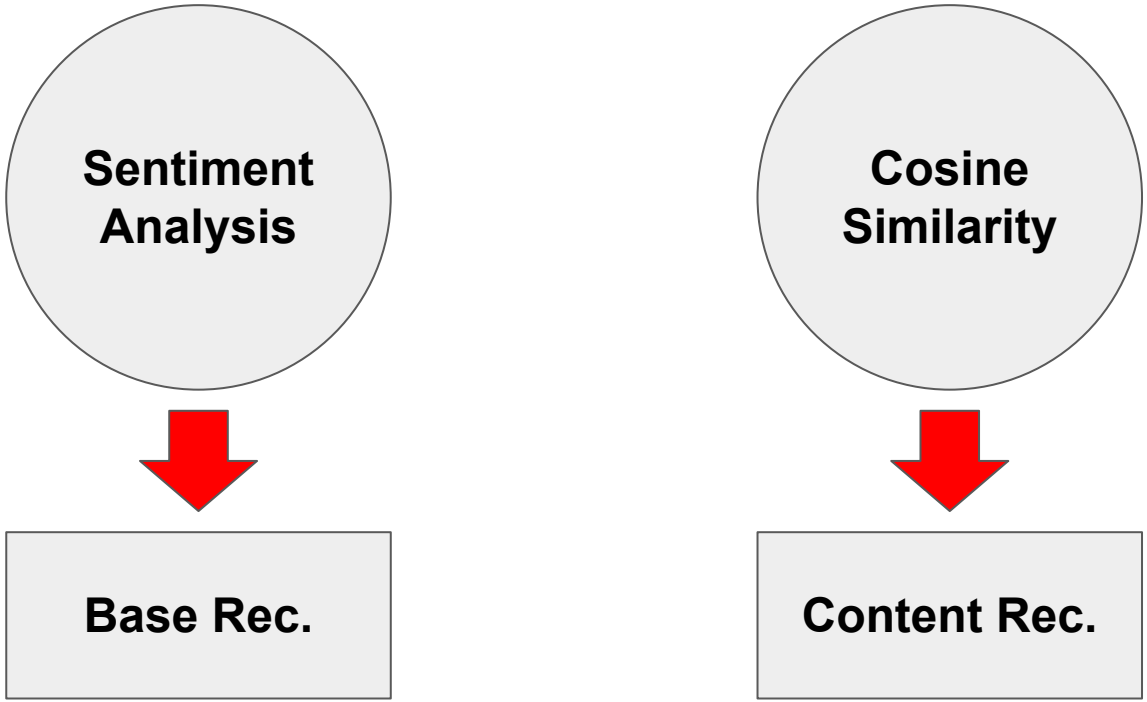
Word Algebra: Examples

'filet_mignon' - 'beef' + 'seafood' = lobster tail

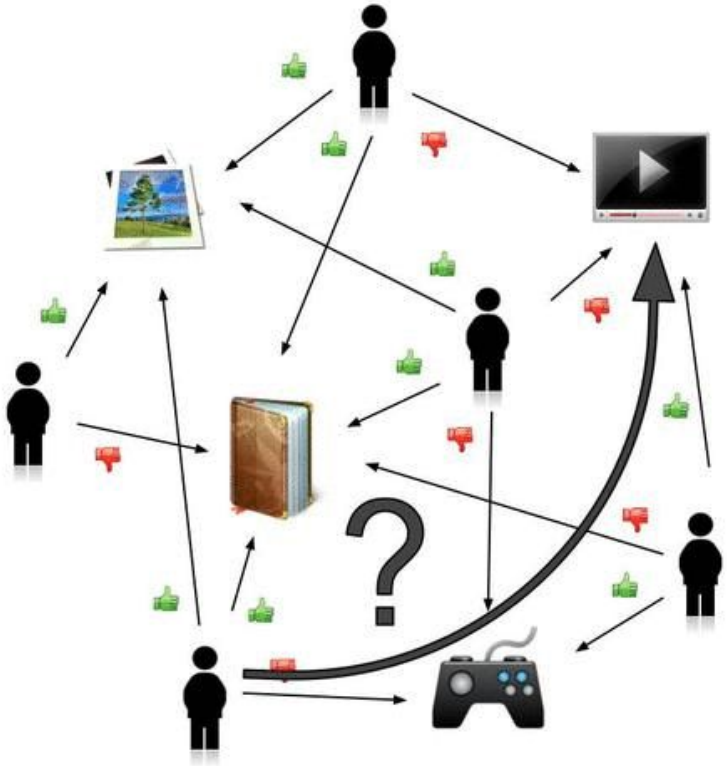
'fork' + 'soup' = spoon

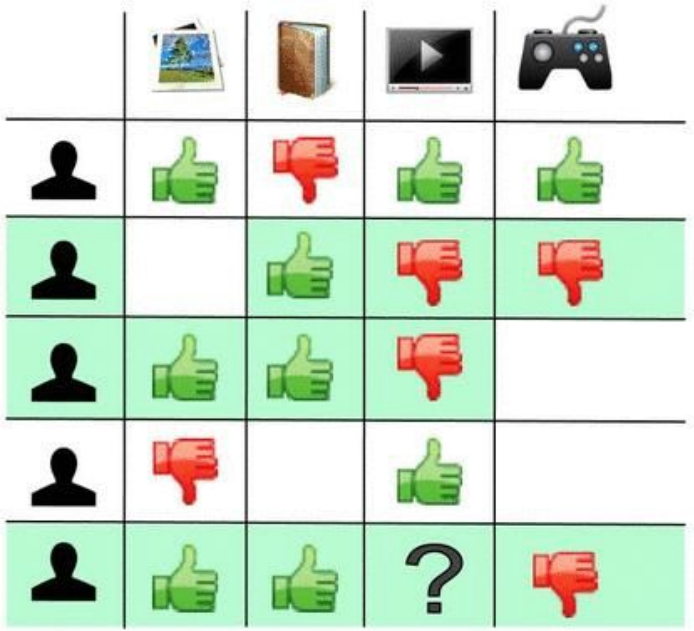
'drink' + 'barley' = beer

NLP Summary


Sentiment Analysis
Base Rec.
Cosine Similarity
Content Rec.

Collaborative Filtering 1

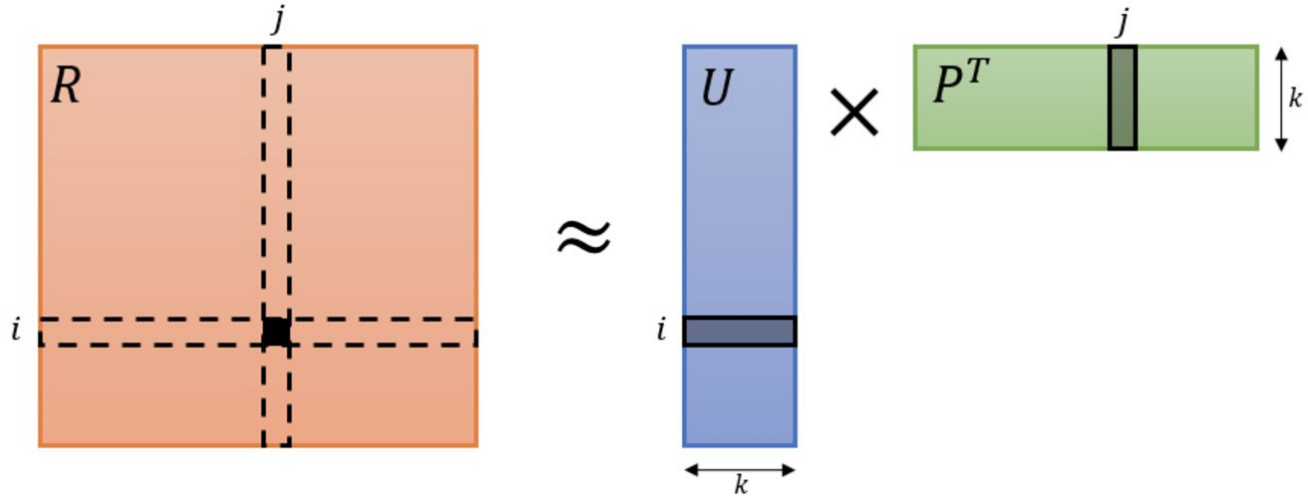




https://en.wikipedia.org/wiki/Collaborative_filtering

Collaborative Filtering 2

Matrix Factorization



Cost Function

$$J = ||R - U \times P^T||_2 + \lambda \left(||U||_2 + ||P||_2 \right)$$

Yelp Social Network 1

nodeDF

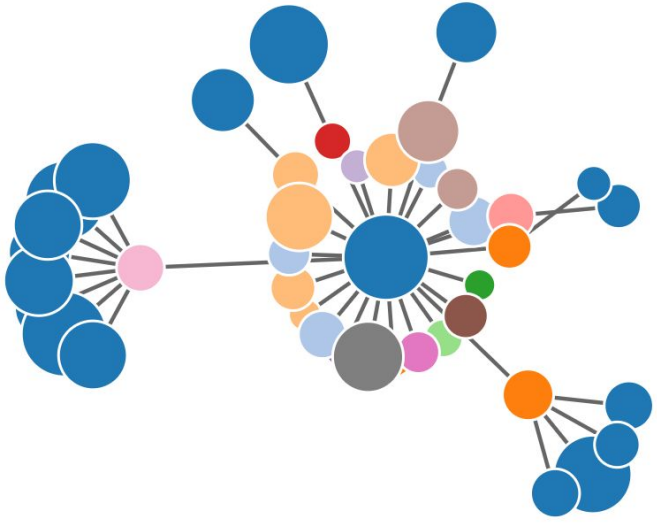
yelp_academic_dataset_user.json

```
{
    "user_id":"encrypted user id",
    "name":"first name",
    "review_count":number of reviews,
    "yelping_since": date formatted like "2009-12-19",
    "friends":["an array of encrypted ids of friends"],
    "useful":"number of useful votes sent by the user",
    "funny":"number of funny votes sent by the user",
    "cool":"number of cool votes sent by the user",
    "fans":"number of fans the user has",
```

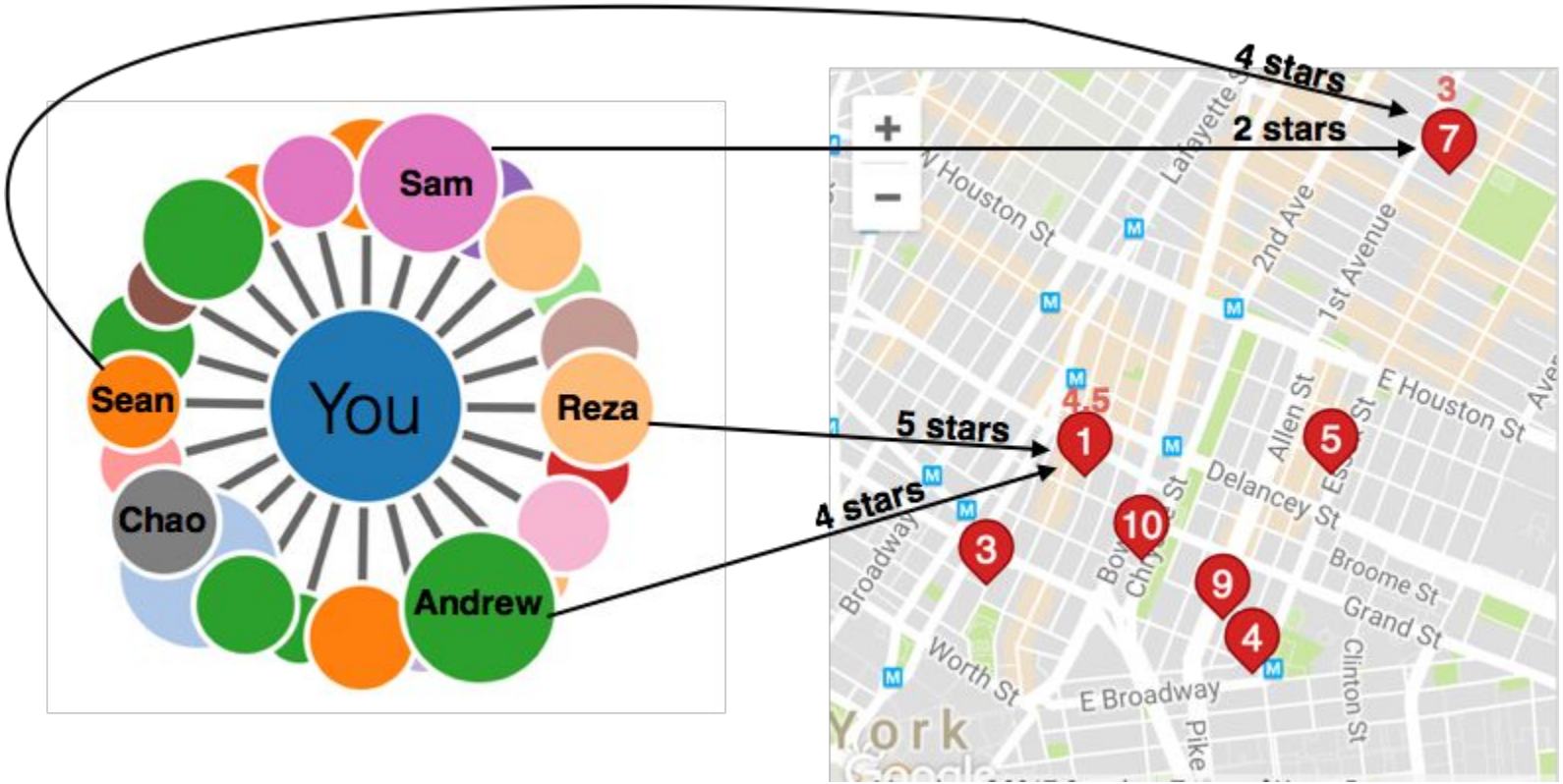
edgeDF

```
friends.show(50)
```

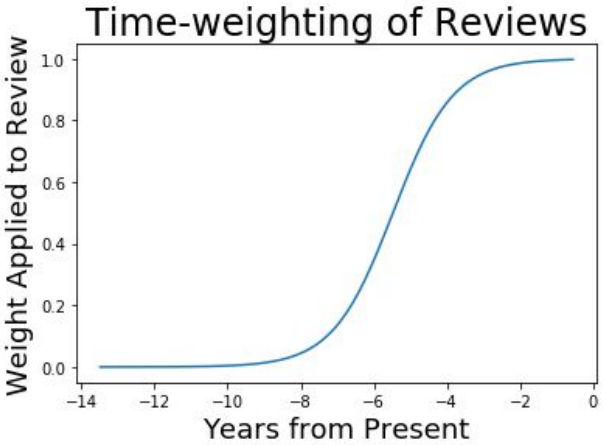
```
+-------+-------+
|     u0|     u1|
+-------+-------+
|1003979|1024458|
| 297574|1003979|
| 970343|1003979|
| 320810|1003979|
| 188106|1003979|
| 492231|1003979|
| 352654|1003979|
```



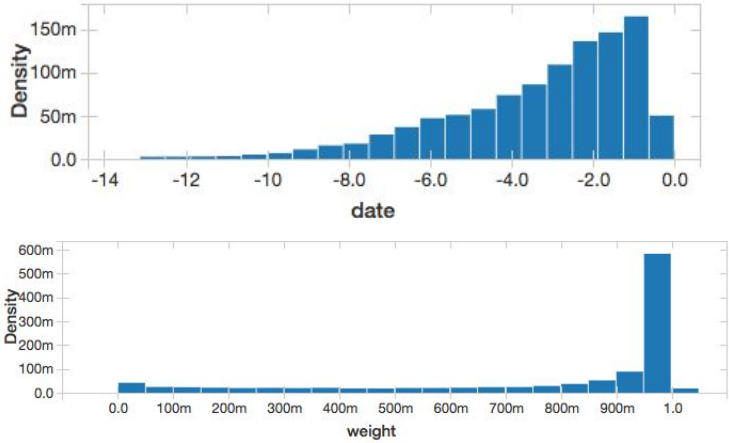
Yelp Social Network 2



Review-level adjustments



Reviews binned by years from present

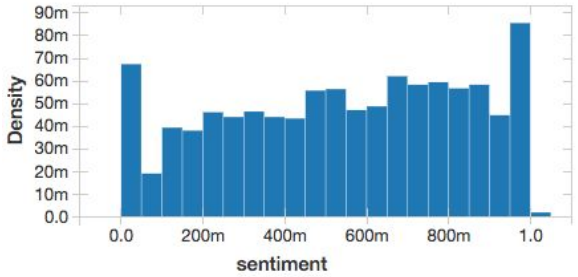


Newest 60% of reviews unaffected by time-weighting

How relevant is a 10 year old review?

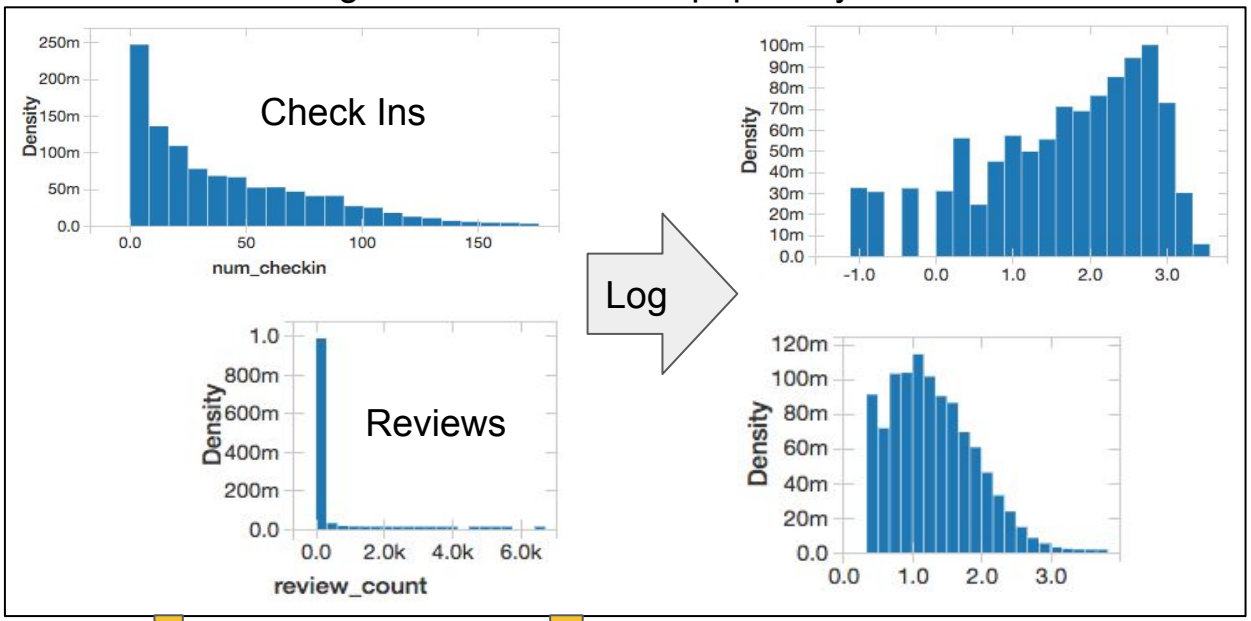
Business-level adjustments

Multiplicative factor based on NLP sentiment



$$\xi = \left(\frac{3}{4} + \frac{\text{sentiment}}{4}\right) \cdot \sum_{\text{all reviews}} f(t, \text{rating})$$

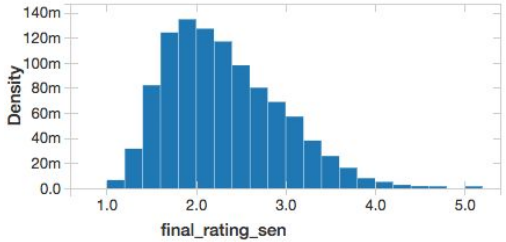
Log transformations for popularity measures

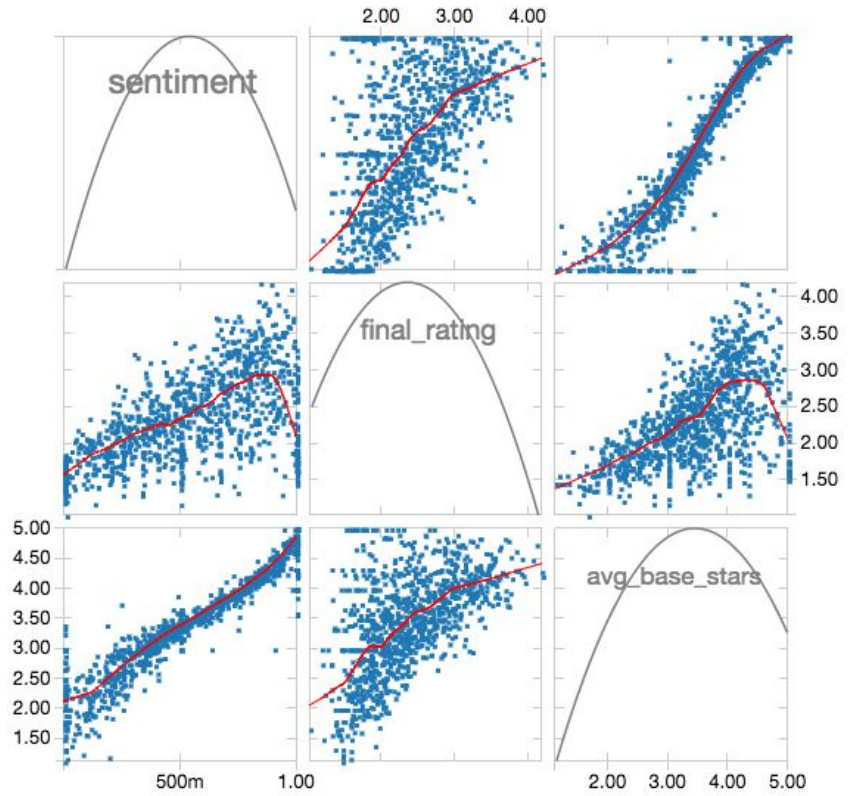


$$\text{final score} = \log_{10}\left(\text{review count}^{\xi} \cdot g(\text{check-in count})\right)$$

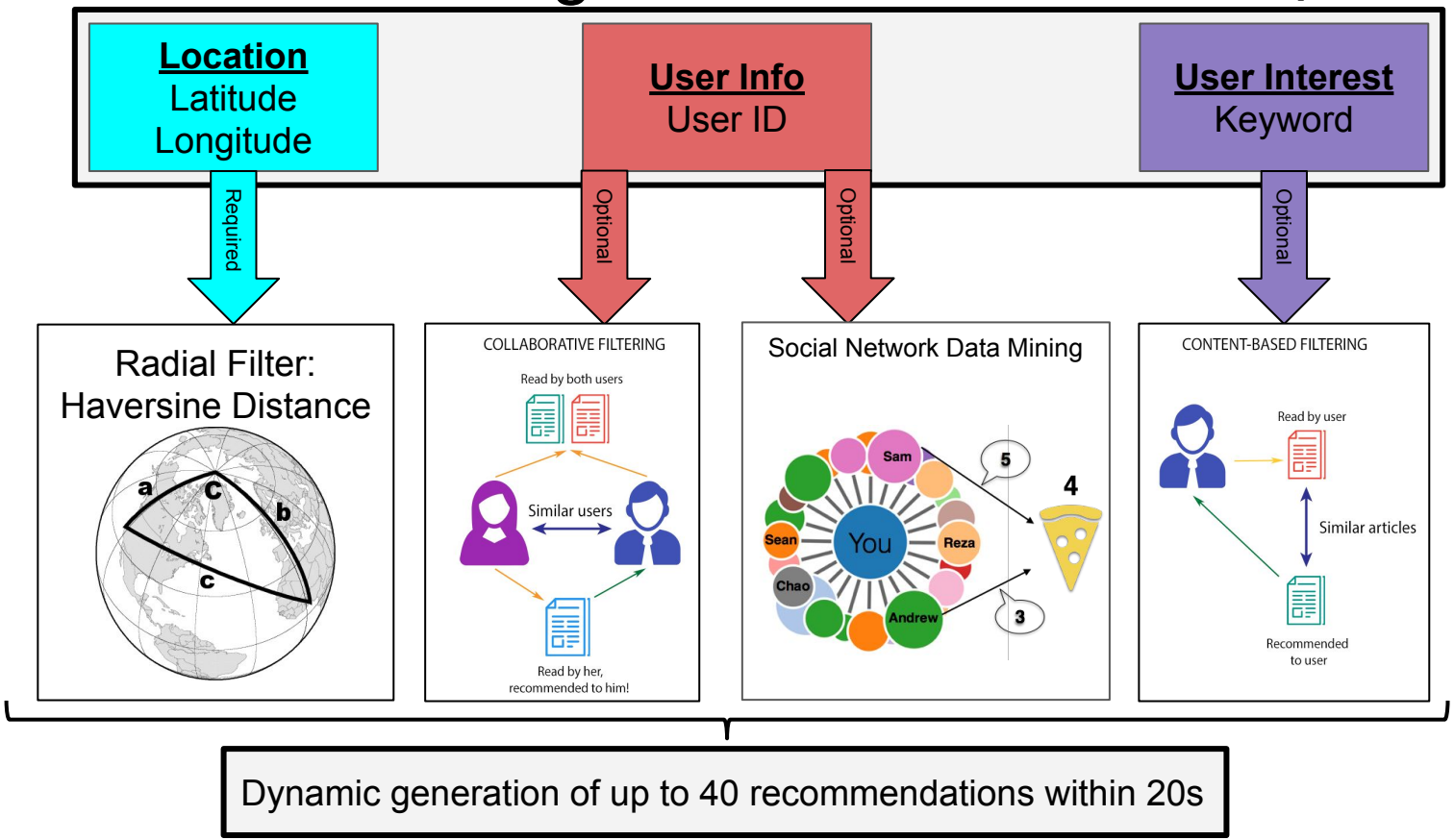
Final Location-only Ratings

Stars	Percentile
★★★★	99th (top 40)
★★★½	95th (top 300)
★★★	83rd
★★½	60th
★★	31st
★½	5th

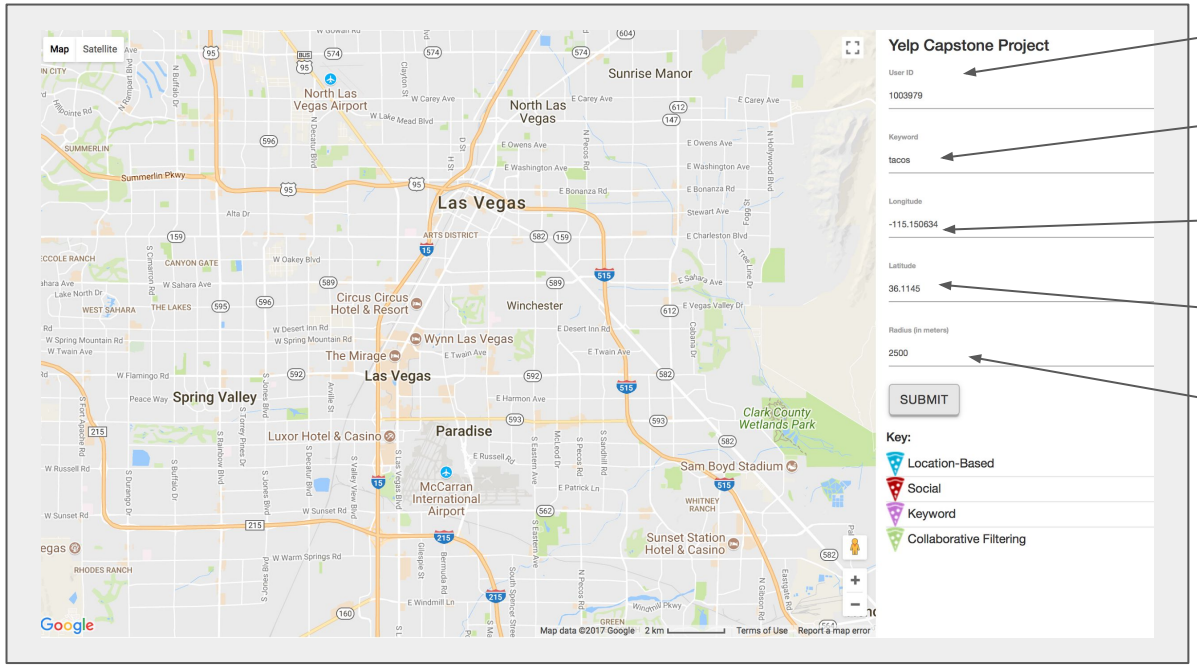




Recommendation Engine



Data Pipeline: 1. Flask



User ID

Keyword

Longitude

Latitude

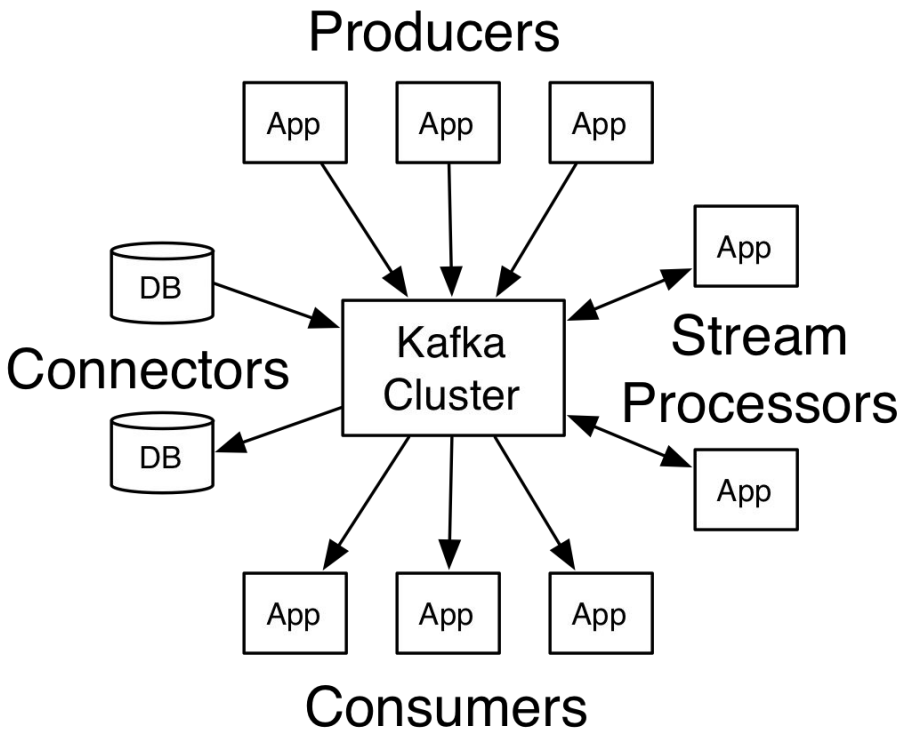
Search Radius

Flask
web development,
one drop at a time

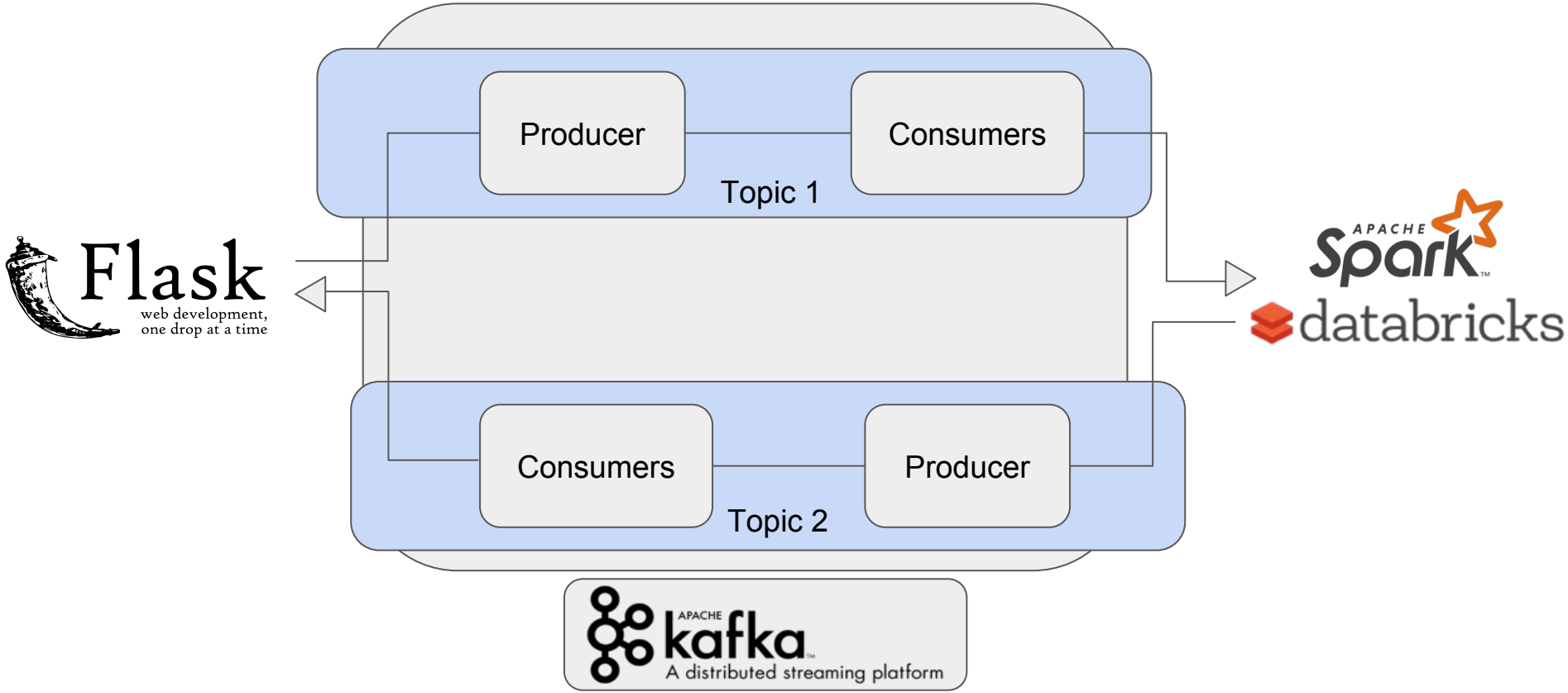
Data Pipeline: Kafka

Components:

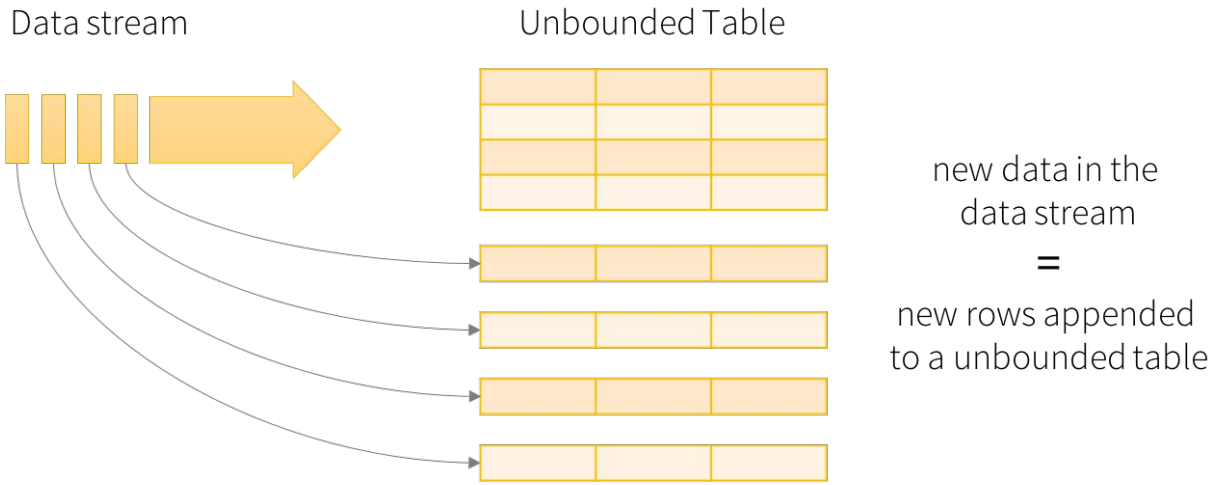
1. Brokers
2. Topics
3. Producers
4. Consumers



Data Pipeline: 2. Kafka



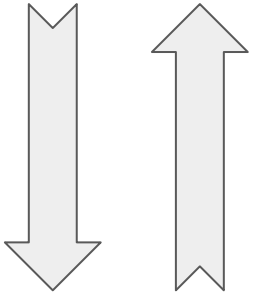
Data Pipeline: 3. Spark

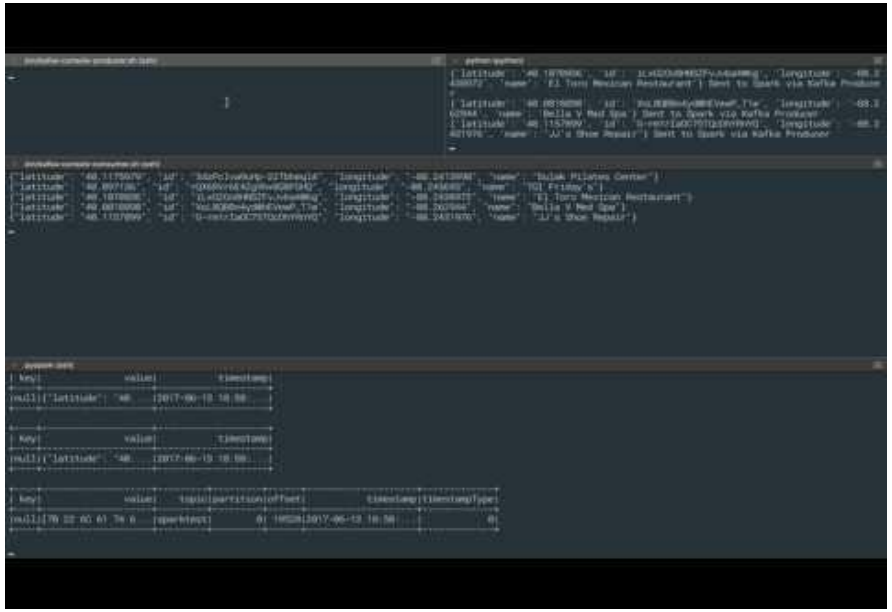


Data stream as an unbounded table

Spark's Structured Streaming

Data Pipeline: Overall





Demo

Lessons Learned and Future Work

Lessons

- Recommendation Algorithms
- Dealing with Big Data
- Working in a Startup-like Environment

Future Work

- Front End and Streaming
  - User experience enhancement
  - Restructure the data streams for better stability
- Back End Algorithms
  - Speed up current code
  - The very fun part has just begun

Acknowledgement

Guidance and Discussion from

- Shu Yan
- Yvonne Lau
- Zeyu Zhang

Inspiration from

Chuan Sun and Aiko Liu

Thanks!