mrjob

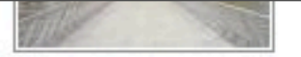
Sudarshan Gaikaiwari
Yelp
http://github.com/Yelp/mrjob

What's the Problem?

- Yelp produces over 100s of GB of logs per day
- Many features rely on log analysis
- Computers are cheap, but leveraging many of them for a single task is hard

Category: Parks [Edit]

The Embarcadero and Mission Street
San Francisco, CA 94105



Add Photos

Good for Kids: Yes

Edit Business Info

First to Review  Toro E.

Send to Friend | Bookmark | Send to Phone | Write a Review | Print

14 reviews for Pier 14

Search Reviews

Sort by: **Yelp Sort** | Date | Rating | Elites' | Facebook Friends'

All Reviews

Elite '10
67
477
Kylie L.
San Francisco, CA

8/6/2010 5 photos

Why is pier 14 http://www.yelp.com/bi... next to pier 2 & not next to pier 15?

The view is AMAZING, especially on a sunny day. Great views of the Bay Bridge http://www.yelp.com/bi..., Port of SF, Coit Tower http://www.yelp.com/bi..., kayakers, boaters, sailors http://www.yelp.com/bi..., & people fishing http://www.yelp.com/bi.... There are individual rotating seats along the pier, which I think is really cool.

View Larger Map/Directions »

Browse Nearby:

Restaurants | Nightlife | Shopping | Movies | All

People Who Viewed This Also Viewed...

Fort Point
137 reviews
San Francisco, CA

Pier 7
10 reviews
Neighborhood: Embarcadero

Mission Creek Park
20 reviews
Neighborhood: SOMA

Pier 37
1 review
Neighborhood: Financial District

Ads by CityGrid

ous Workout. Small Classes.
com/?scid=1830115

Search Reviews

er 15?

Bay Bridge
/bi..., kayakers, boaters,
m/bi.... There are individual

View Larger Map/Directions »

Browse Nearby:

Restaurants | Nightlife | Shopping | Movies | All

People Who Viewed This Also Viewed...

Fort Point
137 reviews
San Francisco, CA

Pier 7
10 reviews
Neighborhood: Embarcadero

Mission Creek Park
20 reviews
Neighborhood: SOMA

Pier 37
1 review
Neighborhood: Financial District

MapReduce

MapReduce: Simplified Data Processing on Large Clusters

Jeffrey Dean and Sanjay Ghemawat

jeff@google.com, sanjay@google.com

Google, Inc.

Abstract

MapReduce is a programming model and an associated implementation for processing and generating large data sets. Users specify a *map* function that processes a key/value pair to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. Many real world tasks are expressible in this model, as shown in the paper.

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The run-time system takes care of the details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication. This allows programmers without any experience with parallel and distributed systems to easily utilize the resources of a large distributed system.

given day, etc. Most such computations are conceptually straightforward. However, the input data is usually large and the computations have to be distributed across hundreds or thousands of machines in order to finish in a reasonable amount of time. The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues.

As a reaction to this complexity, we designed a new abstraction that allows us to express the simple computations we were trying to perform but hides the messy details of parallelization, fault-tolerance, data distribution and load balancing in a library. Our abstraction is inspired by the *map* and *reduce* primitives present in Lisp and many other functional languages. We realized that most of our computations involved applying a *map* operation to each logical "record" in our input in order to compute a set of intermediate key/value pairs, and then applying a *reduce* operation to all the values that shared

Go, Dog. Go!

go
dog
go

go ⟶ counter() ⟶ (g, 1), (o, 1)

dog ⟶ counter() ⟶ (d, 1), (o, 1), (g, 1)

go ⟶ counter() ⟶ (g, 1), (o, 1)

(g, 1), (o, 1)

(d, 1), (o, 1), (g, 1) ⟶ sort ⟶

(g, 1), (o, 1)

(d, 1)
(g, 1)
(g, 1)
(g, 1)
(o, 1)
(o, 1)
(o, 1)

(d, 1)

(g, 1)

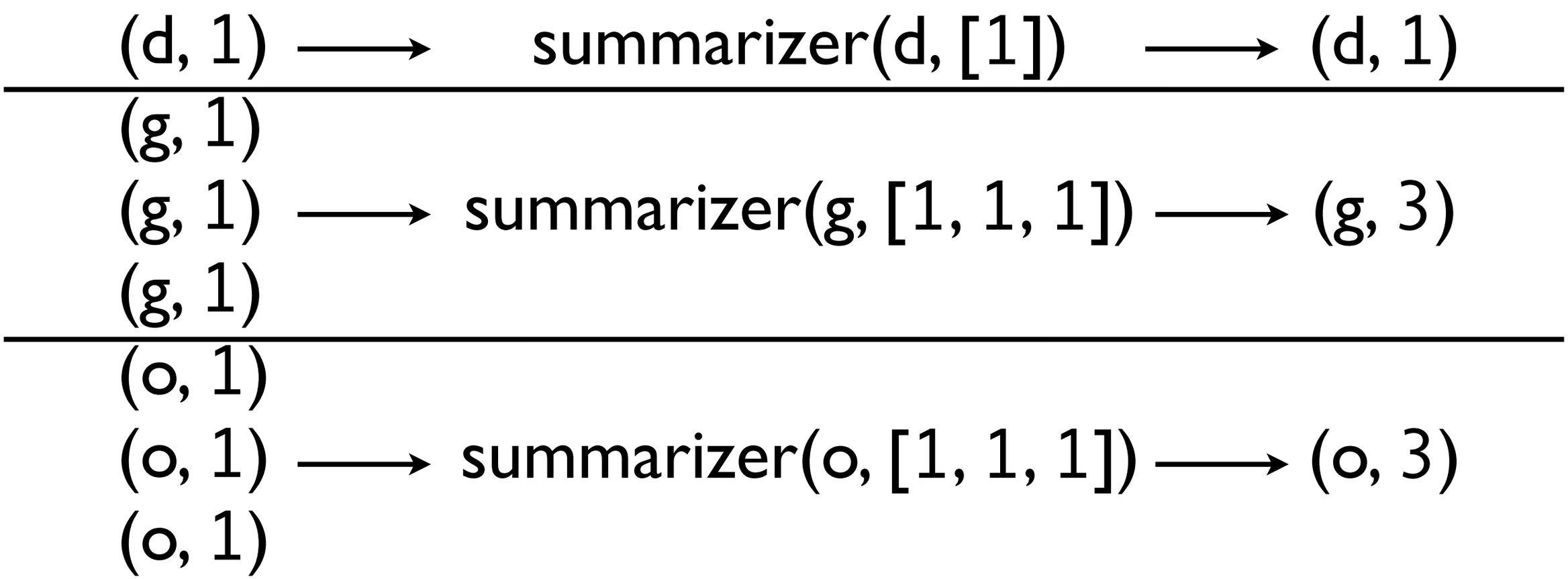
(g, 1)

(g, 1)

(o, 1)

(o, 1)

(o, 1)


(d, 1)
summarizer(d, [1])
(d, 1)
(g, 1)
(g, 1)
(g, 1)
summarizer(g, [1, 1, 1])
(g, 3)
(g, 1)
(o, 1)
(o, 1)
(o, 1)
summarizer(o, [1, 1, 1])
(o, 3)
(o, 1)

go ⟶ counter() ⟶ (g, 1), (o, 1)

dog ⟶ counter() ⟶ (d, 1), (o, 1), (g, 1)

go ⟶ counter() ⟶ (g, 1), (o, 1)

go ⟶ counter() ⟶ (g, 1), (o, 1)

dog ⟶ counter() ⟶ (d, 1), (o, 1), (g, 1)

go ⟶ counter() ⟶ (g, 1), (o, 1)

go ⟶ mapper() ⟶ (g, 1), (o, 1)

dog ⟶ mapper() ⟶ (d, 1), (o, 1), (g, 1)

go ⟶ mapper() ⟶ (g, 1), (o, 1)

(d, 1) ⟶ summarizer(d, [1]) ⟶ (d, 1)

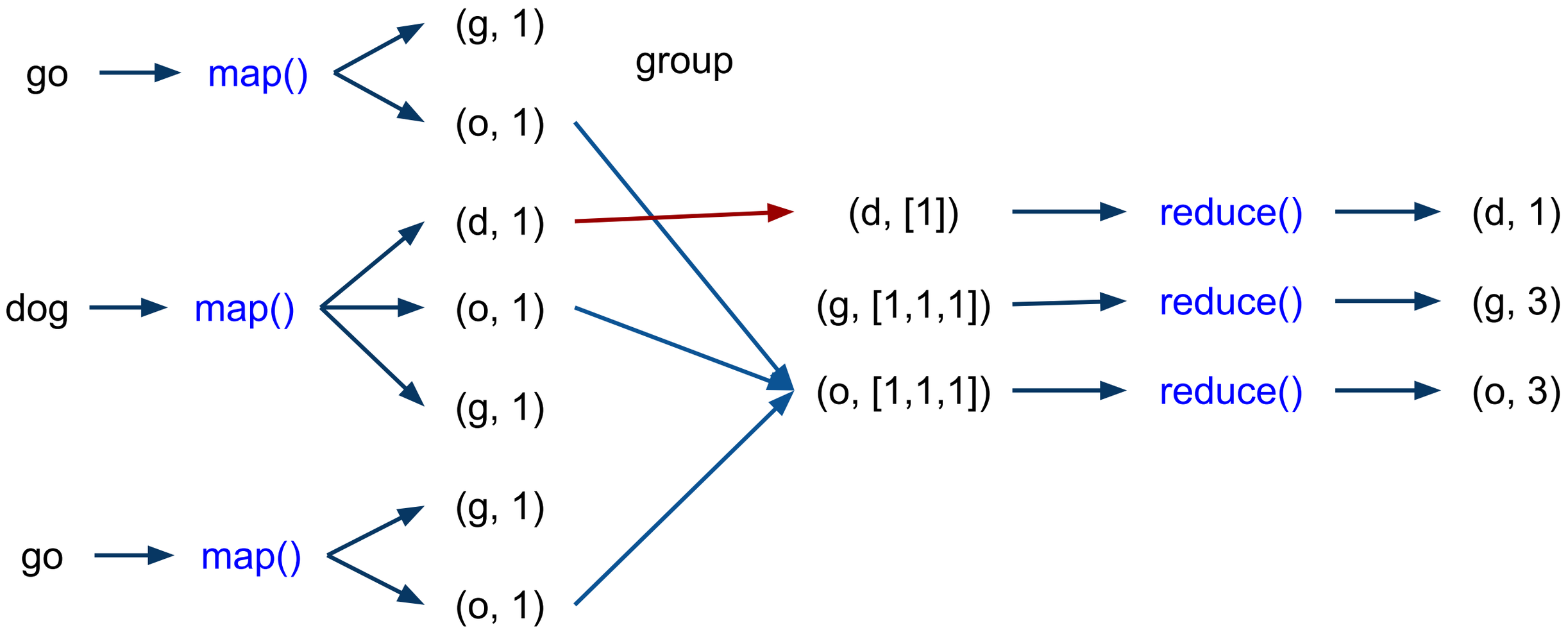
(g, 1)
(g, 1) ⟶ summarizer(g, [1, 1, 1]) ⟶ (g, 3)
(g, 1)

(o, 1)
(o, 1) ⟶ summarizer(o, [1, 1, 1]) ⟶ (o, 3)
(o, 1)

(d, 1) ⟶ reducer(d, [1]) ⟶ (d, 1)

(g, 1)
(g, 1) ⟶ reducer(g, [1, 1, 1]) ⟶ (g, 3)
(g, 1)

(o, 1)
(o, 1) ⟶ reducer(o, [1, 1, 1]) ⟶ (o, 3)
(o, 1)


go
map()
(g, 1)
(o, 1)
dog
map()
(d, 1)
(o, 1)
(g, 1)
go
map()
(g, 1)
(o, 1)
group
(d, [1])
(g, [1,1,1])
(o, [1,1,1])
reduce()
reduce()
reduce()
(d, 1)
(g, 3)
(o, 3)

Yelp's mrjob

- sign up for Amazon Elastic MapReduce
- write a dozen lines of Python
- Test locally on your machine
- run on many Amazon computers

```
from mrjob.job import MRJob

class MRCharacterCount(MRJob):
    def mapper(self, _, text):
        for c in text:
            yield c, 1

    def reducer(self, c, counts):
        yield c, sum(counts)

if __name__ == '__main__':
    MRCharacterCount.run()
```

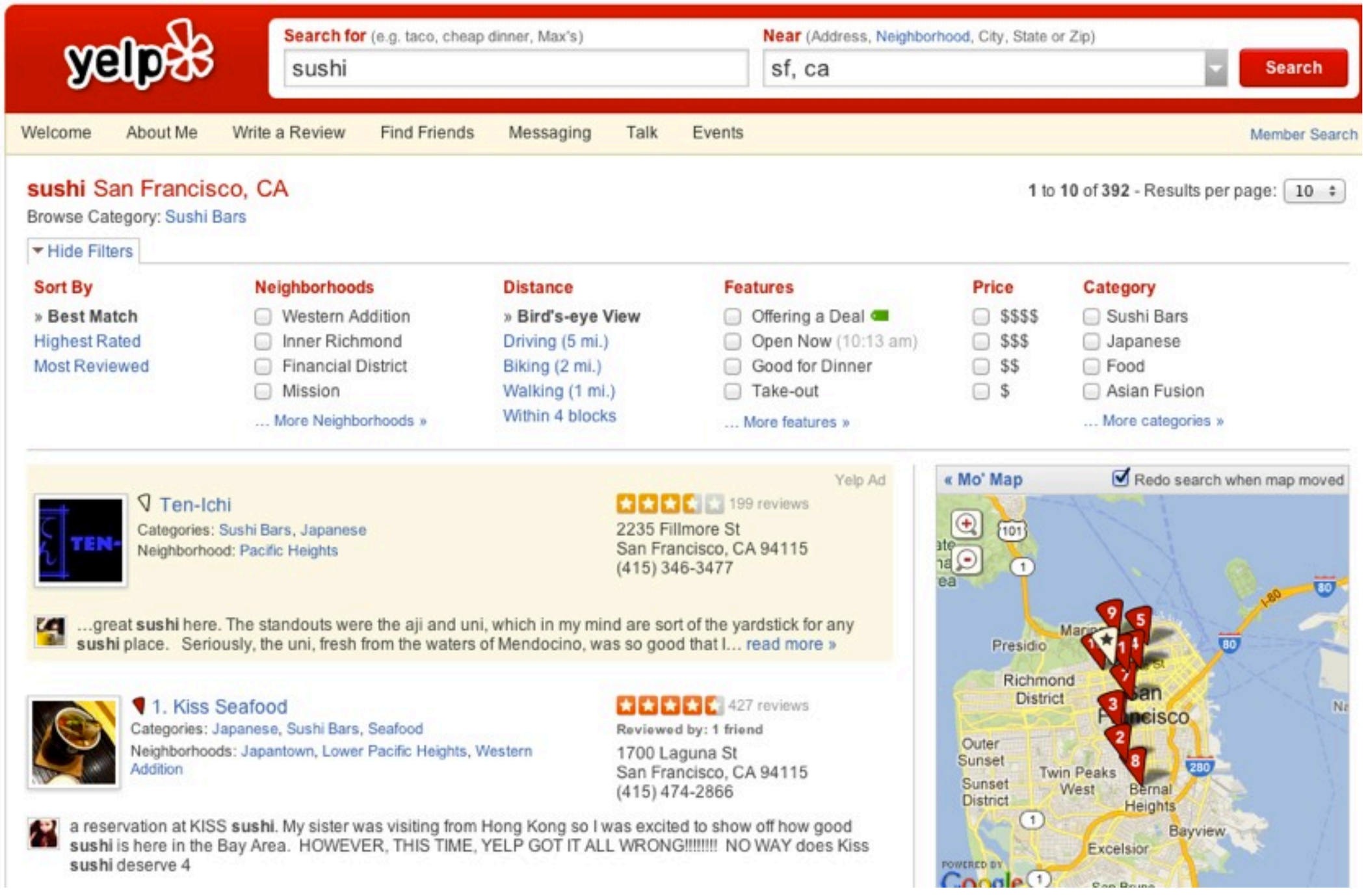
```
from mrjob.job import MRJob

class MRWordCount(MRJob):
    def mapper(self, _, text):
        for word in text.split():
            yield word, 1

    def reducer(self, word, counts):
        yield word, sum(counts)

if __name__ == '__main__':
    MRWordCount.run()
```

Ads CTR example



CTR

$$CTR = \frac{Clicks}{Impressions}$$

$$= \frac{Clicks}{Clicks + Did\ Not\ Click}$$

Basic CTR

- https://github.com/sudarshang/mrjob_presentation_code/blob/master/ctr.py

CTR by Ad Campaign

- https://github.com/sudarshang/mrjob_presentation_code/blob/master/ctr_by_campaign.py

CTR by Campaign and type

- https://github.com/sudarshang/mrjob_presentation_code/blob/master/ctr_by_campaign_and_type.py

CTR by campaign types and totals

- https://github.com/sudarshang/mrjob_presentation_code/blob/master/ctr_by_campaign_and_type_with_totals.py

CTR Fatigue - I

- https://github.com/sudarshang/mrjob_presentation_code/blob/master/ctr_fatigue_step_1.py

CTR Fatigue

- https://github.com/sudarshang/mrjob_presentation_code/blob/master/ctr_fatigue.py

Supported Hadoop Features

- Combiners
- Partitioners
- Custom File Formats

Where can I get cool data to play with?

Where can I get cool data to play with?

Yelp is hiring!

yelp.com/jobs