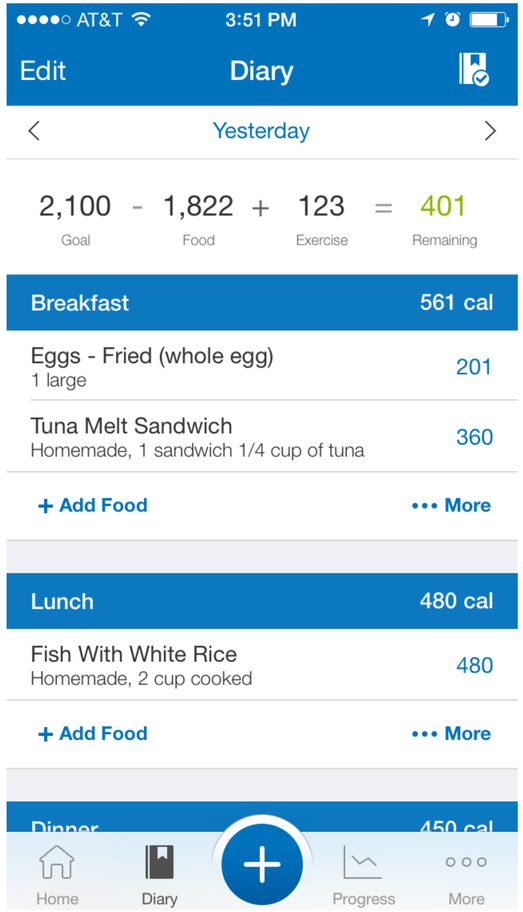
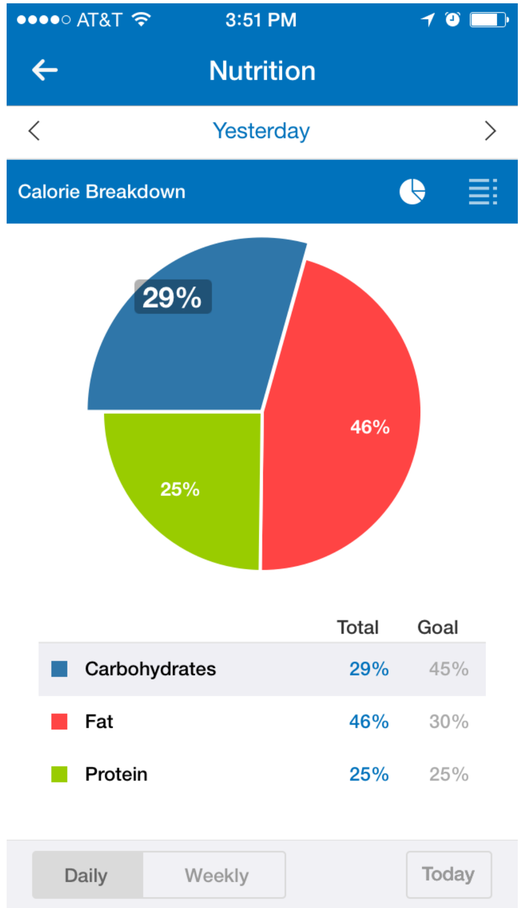
A Spark Based Data Pipeline to Construct a Reliable Food Dataset

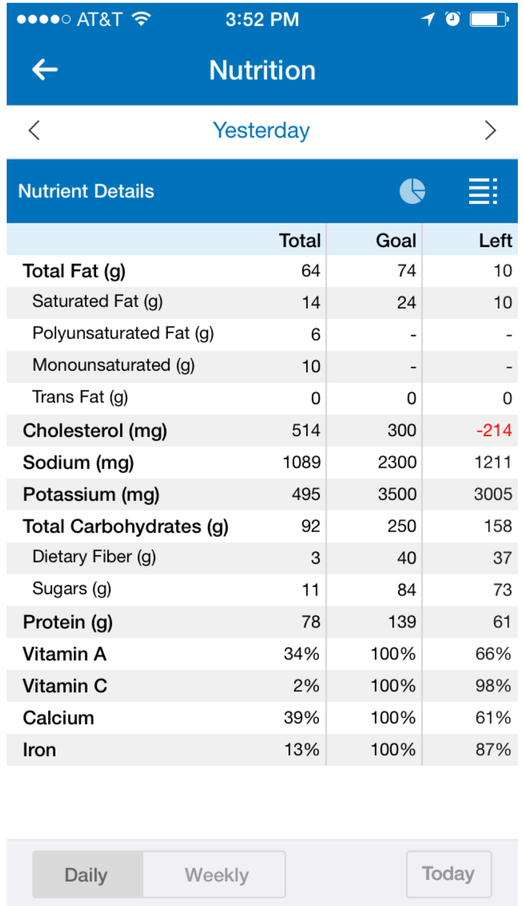
Hesamoddin Salehian

Data Scientist / MyFitnessPal

MyFitnessPal is a free phone app and website to log and track foods/exercises







	Total	Goal	Left
Total Fat (g)	64	74	10
Saturated Fat (g)	14	24	10
Polyunsaturated Fat (g)	6	-	-
Monounsaturated (g)	10	-	-
Trans Fat (g)	0	0	0
Cholesterol (mg)	514	300	-214
Sodium (mg)	1089	2300	1211
Potassium (mg)	495	3500	3005
Total Carbohydrates (g)	92	250	158
Dietary Fiber (g)	3	40	37
Sugars (g)	11	84	73
Protein (g)	78	139	61
Vitamin A	34%	100%	66%
Vitamin C	2%	100%	98%
Calcium	39%	100%	61%
Iron	13%	100%	87%

Some Stats…

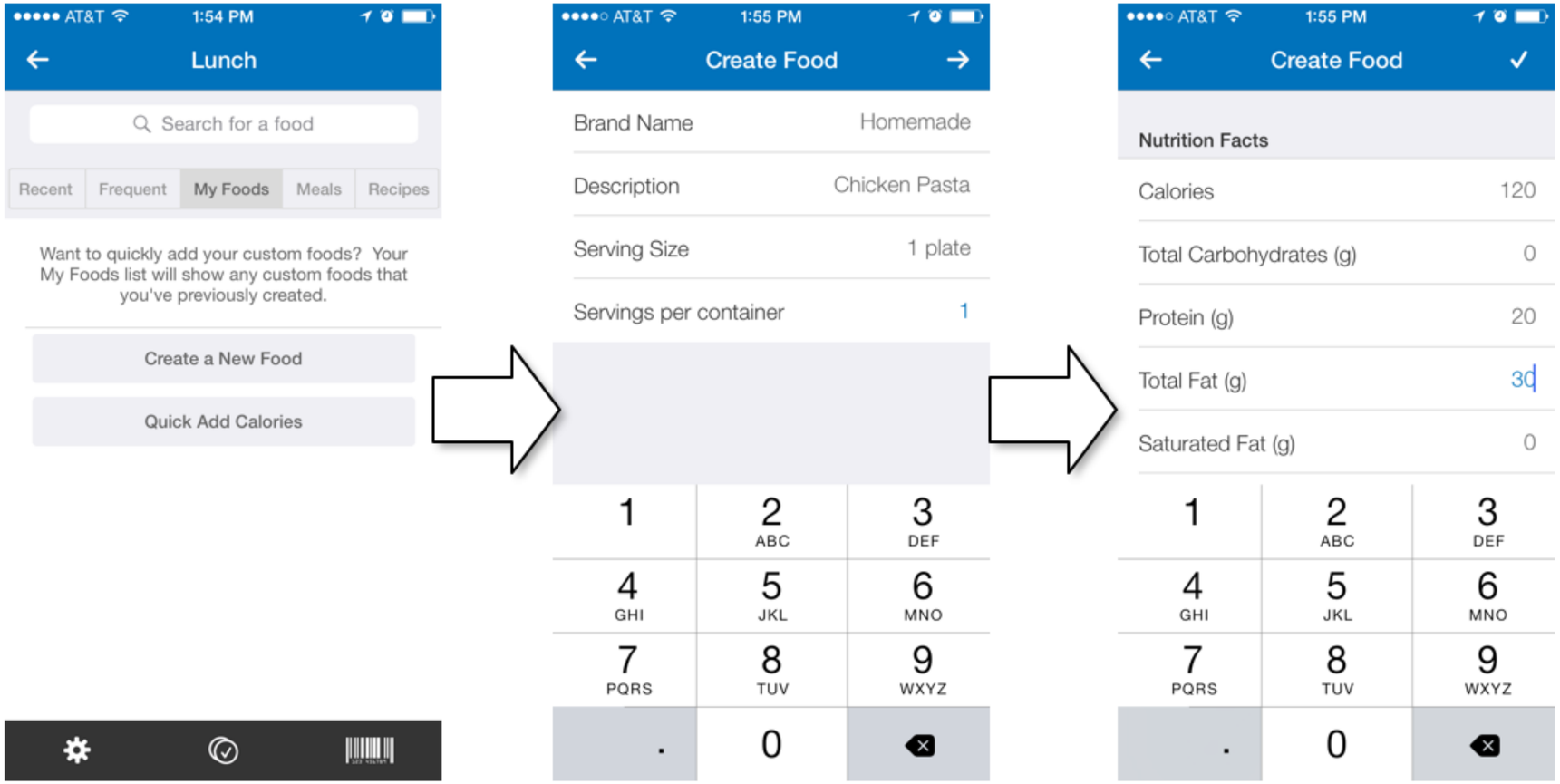
- MyFitnessPal now has more than **80 million** registered users worldwide!

- Collectively, our users have lost over **180 million pound**
  ~= 120,000 elephants!
  ~= 450 blue whales!
- Logged over 14.5 billion foods
- Logged 1.2 Trillion minutes of exercise

  …
- Helped to grow our food database to more than **5 million**!

Crowd-Sourced Food Data

Users are allowed to enter food data into our universally used food database



Crowd-Sourced Food Data (cont.)

Pros:

- ☑ Helps to fill-in the missing foods in MFP database!
- ☑ Enables users to create their own version of foods!

Cons:

- ☑ Significant amount of inconsistencies!

Issue #1

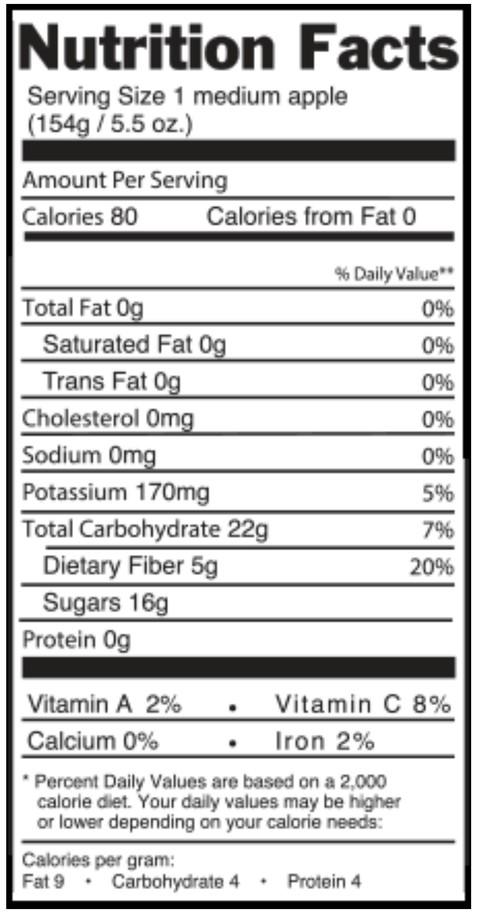
Some food items are represented in multiple free text forms, resulting in duplicates

Example:

"McDonalds McChicken"
"McChicken"
"McDonalds McChicken Sandwich"

Issue #2

Duplicates have incomplete and inconsistent nutritional information



Nutrition Facts

Serving Size 1 medium apple (154g / 5.5 oz.)

Amount Per Serving

Calories 80 Calories from Fat 0

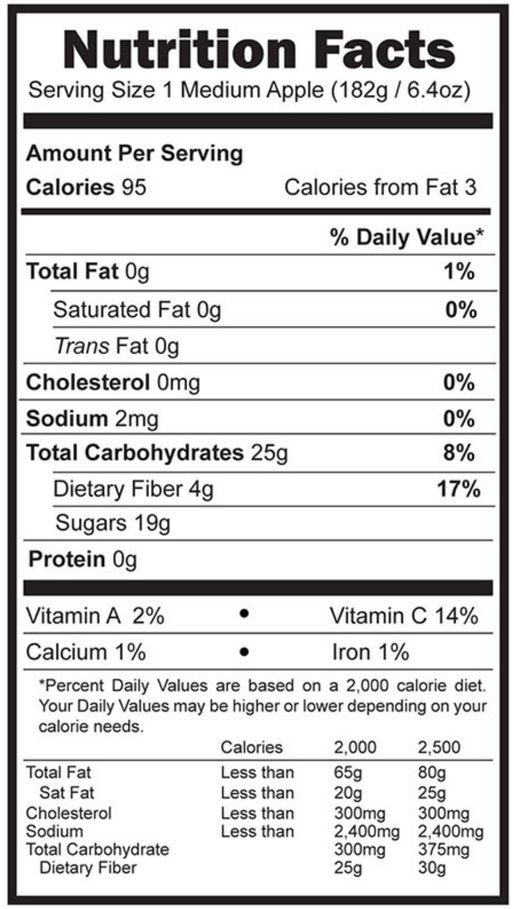
	% Daily Value**
Total Fat 0g	0%
Saturated Fat 0g	0%
Trans Fat 0g	0%
Cholesterol 0mg	0%
Sodium 0mg	0%
Potassium 170mg	5%
Total Carbohydrate 22g	7%
Dietary Fiber 5g	20%
Sugars 16g	
Protein 0g	

Vitamin A 2% • Vitamin C 8%

Calcium 0% • Iron 2%

* Percent Daily Values are based on a 2,000 calorie diet. Your daily values may be higher or lower depending on your calorie needs:

Calories per gram:
Fat 9 • Carbohydrate 4 • Protein 4



Nutrition Facts

Serving Size 1 Medium Apple (182g / 6.4oz)

Amount Per Serving

Calories 95 Calories from Fat 3

	% Daily Value*
Total Fat 0g	**1%**
Saturated Fat 0g	**0%**
Trans Fat 0g	
Cholesterol 0mg	**0%**
Sodium 2mg	**0%**
Total Carbohydrates 25g	**8%**
Dietary Fiber 4g	**17%**
Sugars 19g	
Protein 0g	

Vitamin A 2% • Vitamin C 14%

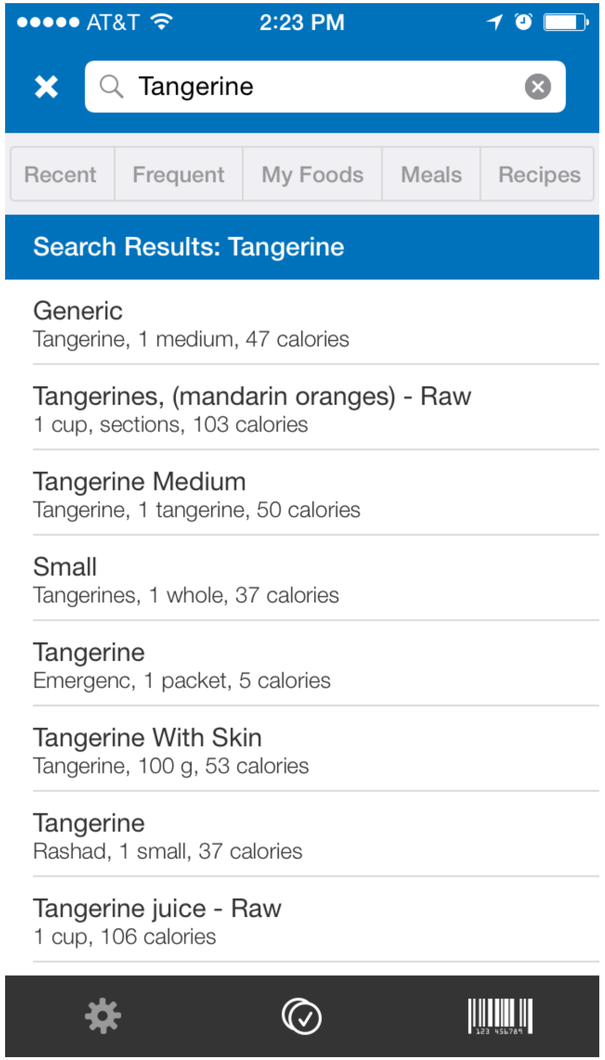
Calcium 1% • Iron 1%

*Percent Daily Values are based on a 2,000 calorie diet. Your Daily Values may be higher or lower depending on your calorie needs.

	Calories	2,000	2,500
Total Fat	Less than	65g	80g
Sat Fat	Less than	20g	25g
Cholesterol	Less than	300mg	300mg
Sodium	Less than	2,400mg	2,400mg
Total Carbohydrate		300mg	375mg
Dietary Fiber		25g	30g

Effects on Food logging

These inconsistencies lead to doubt amongst users regarding the credibility of information and inconvenience at the time of food logging



Solution

Incorporate user endorsement signals in detecting the most accurate subset of foods:

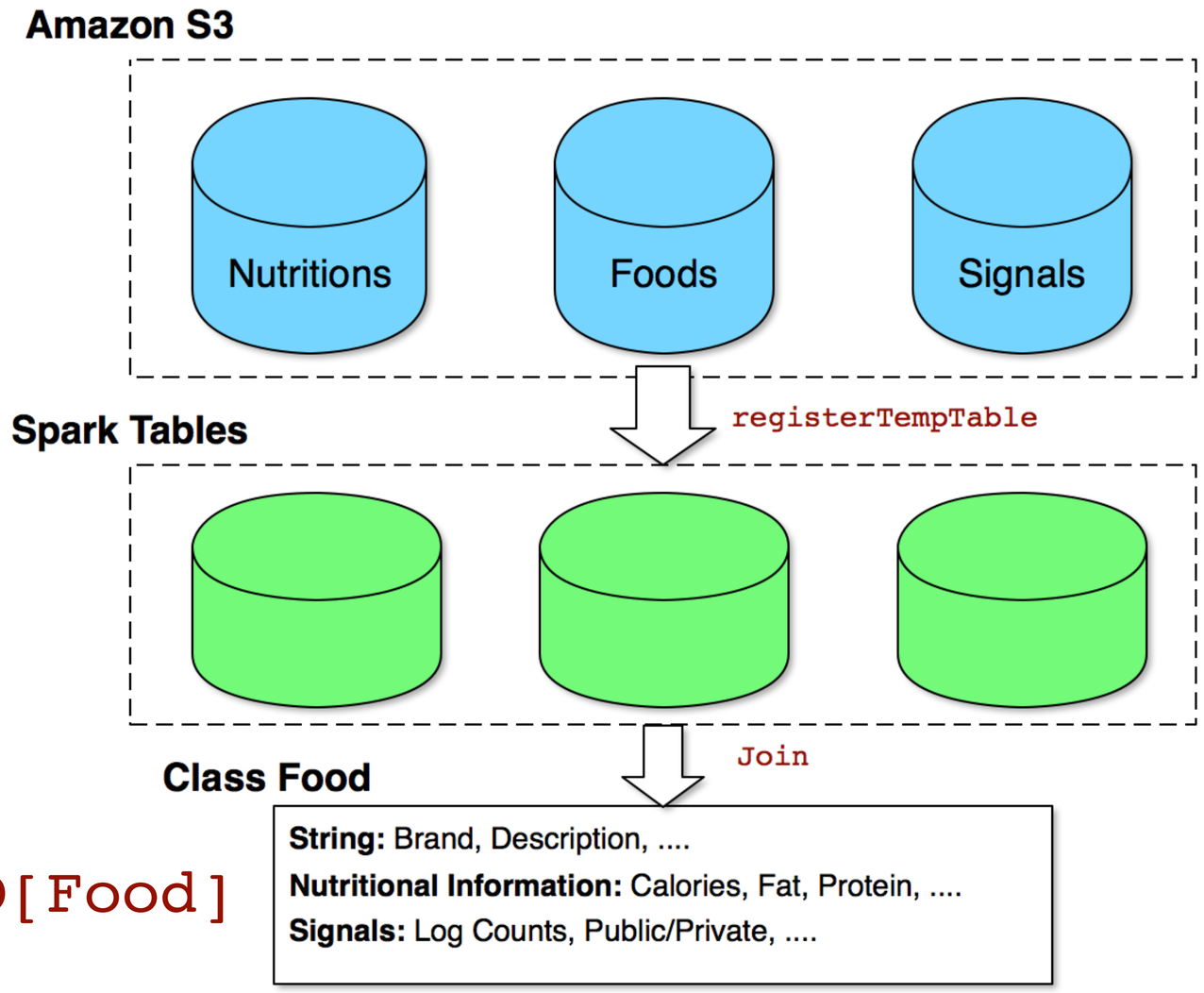
- Implicit (number of logs, number of similar foods created, etc.)
- Explicit (confirmation, public/private, etc.)

… using Spark!

Why Spark?

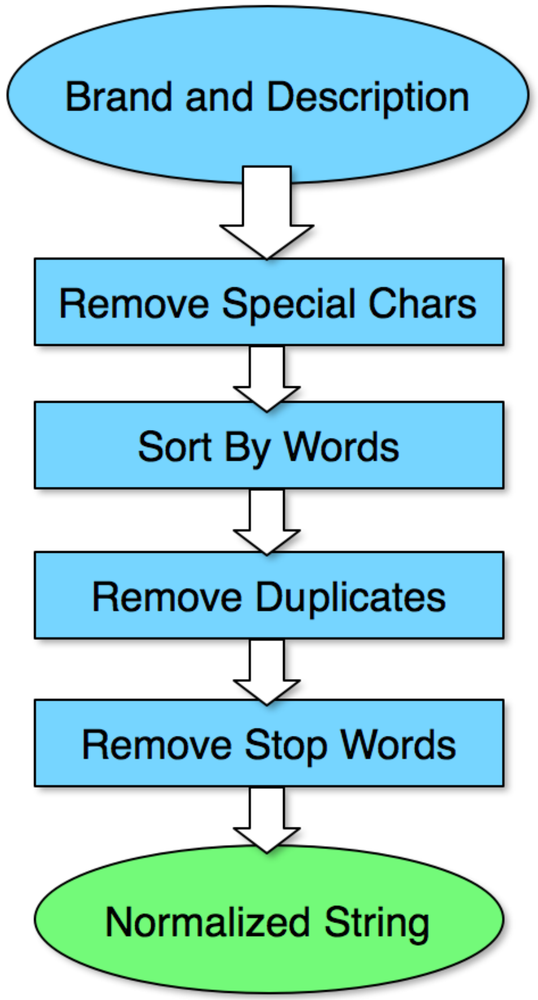
- ☑ Seamless Data Flow Integration between different sources
- ☑ Availability of data processing libraries of MLlib and GraphX
- ☑ Avoid slow offline Table Joins
- ☑ The significantly quicker execution of operations that could be naturally Parallelized

Input Data




```
foods: RDD[Food]
```

String Pre-Processing



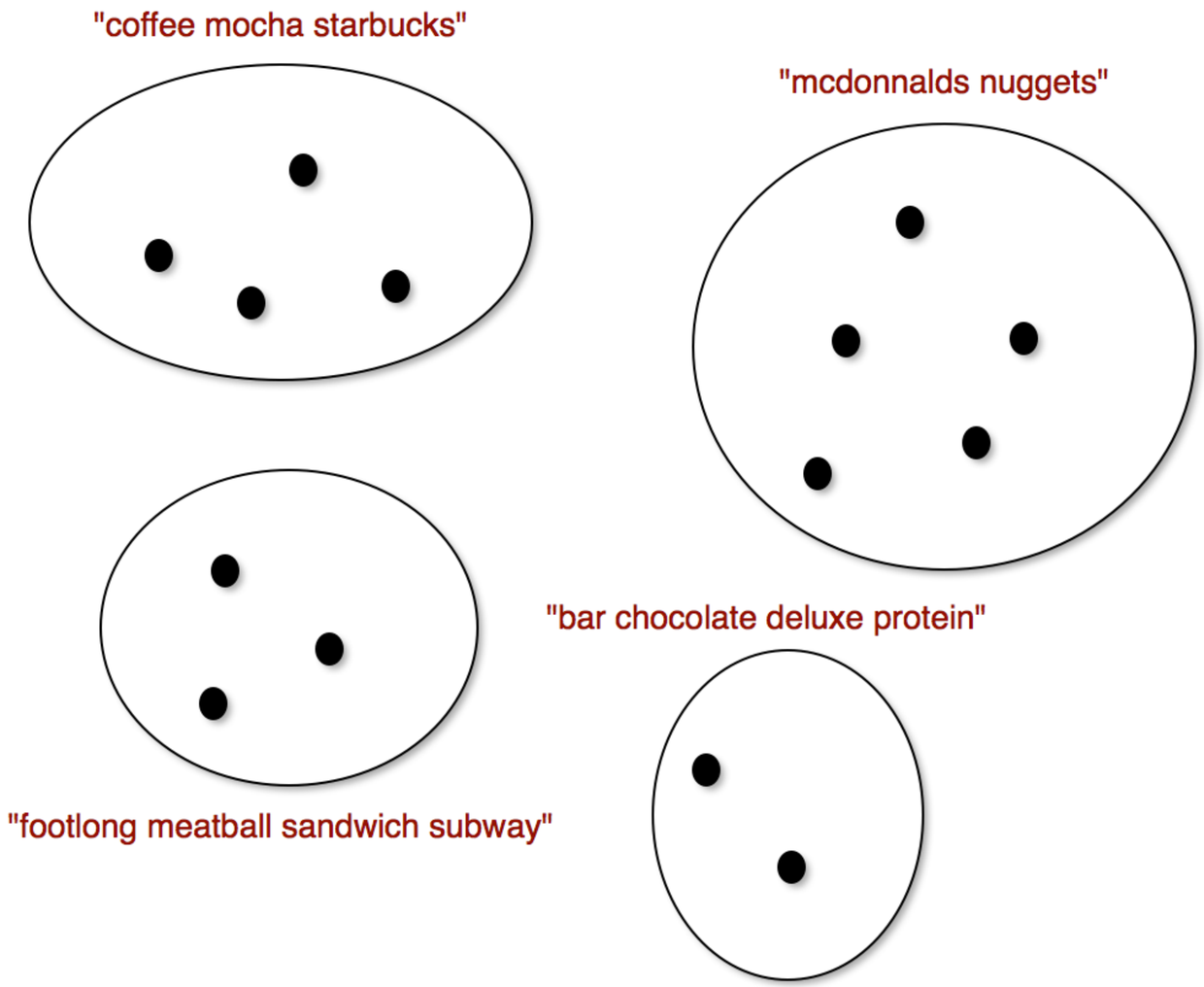
Parallelizable!

Clustering

After normalization the text representation of food items, exact and very-near duplicates are clustered based on textual information:

```
foods.map {case f: Food =>
  (normalize(f.brand + " " + f.description), f)}
  .groupByKey()
```

Clustering

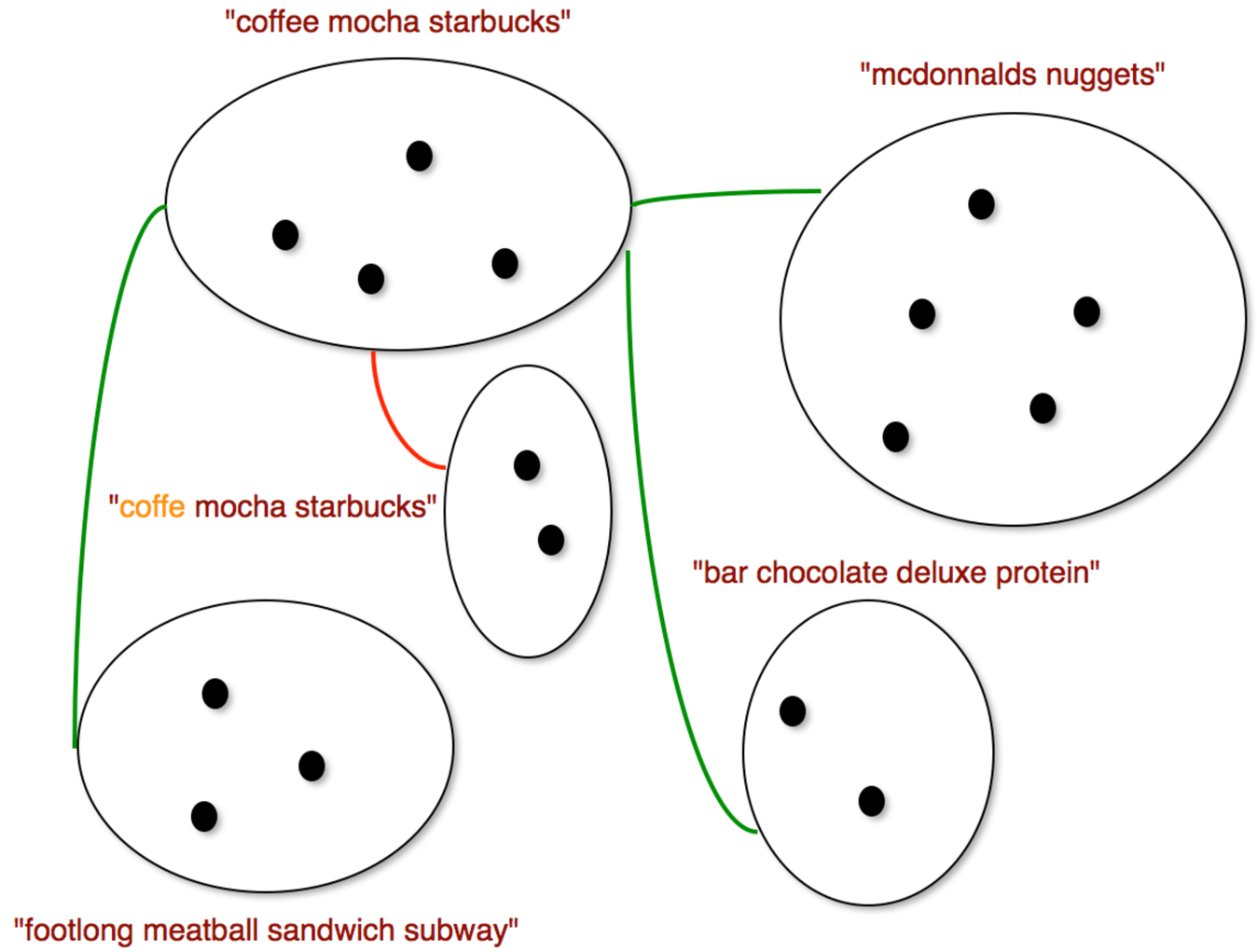


Dedup

Naive Approach: Pair-wise string comparison of the clusters via Edit Distance

```
val pairs =
foods.cartesian(foods)
```

☑ Too slow for the scale of our food database



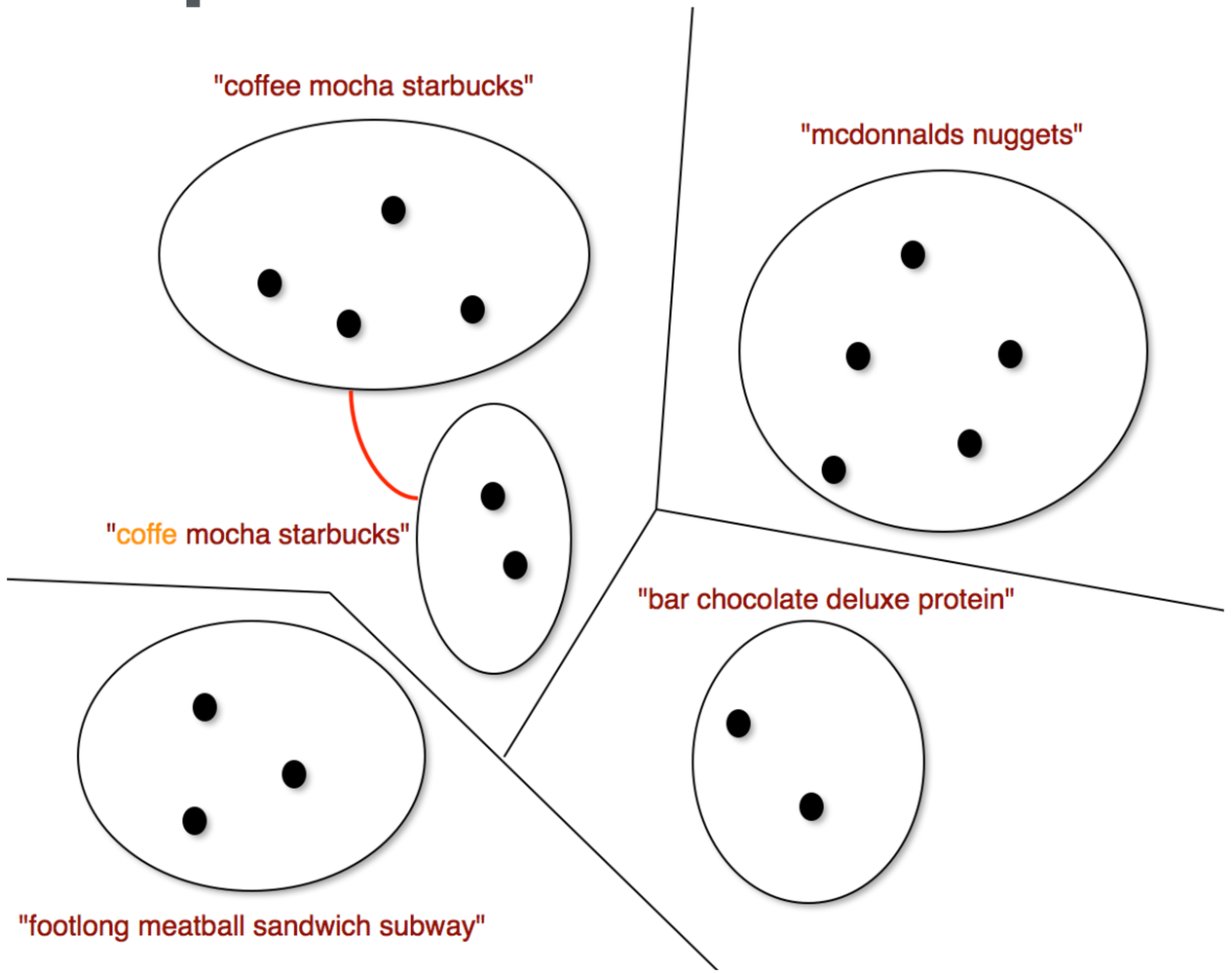
Dedup via LSH

LSH based Approach:

1. Locality Sensitive Hashing on the cluster names to detect duplicate candidates

2. Pair-wise string comparison between candidates only

☑ Significantly decreased the running time!



LSH Code for Spark: https://github.com/mrsqueeze/spark-hash

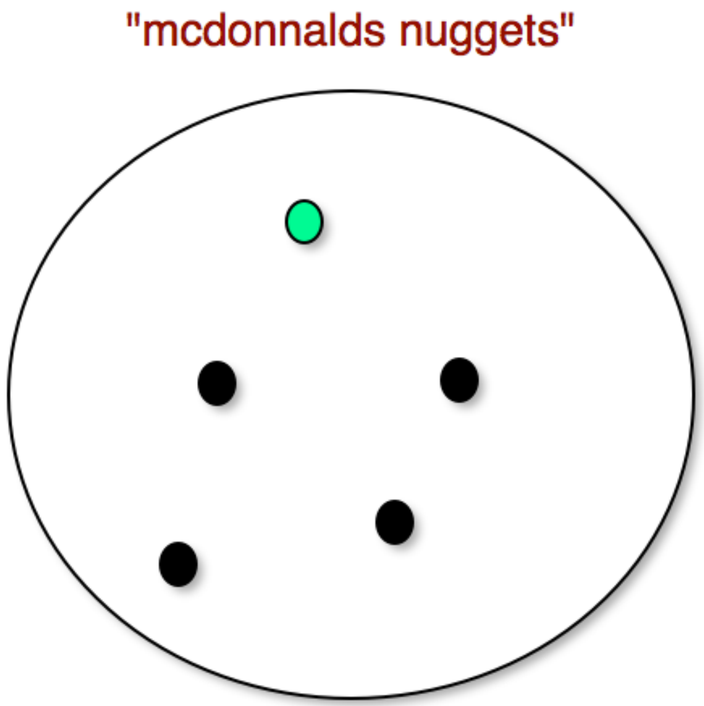
Class Representative Identification

Compute Food Quality Score for each member of cluster,

- Number of times logged
- Number of times confirmed
- Public/private
- …

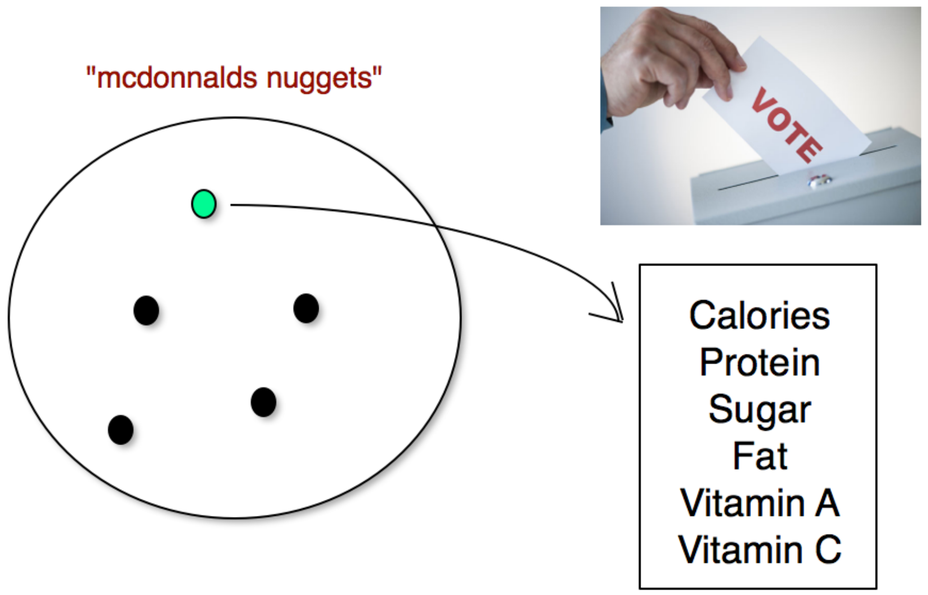
Pick the one with maximum score!

Parallelizable!



Nutrition Aggregation

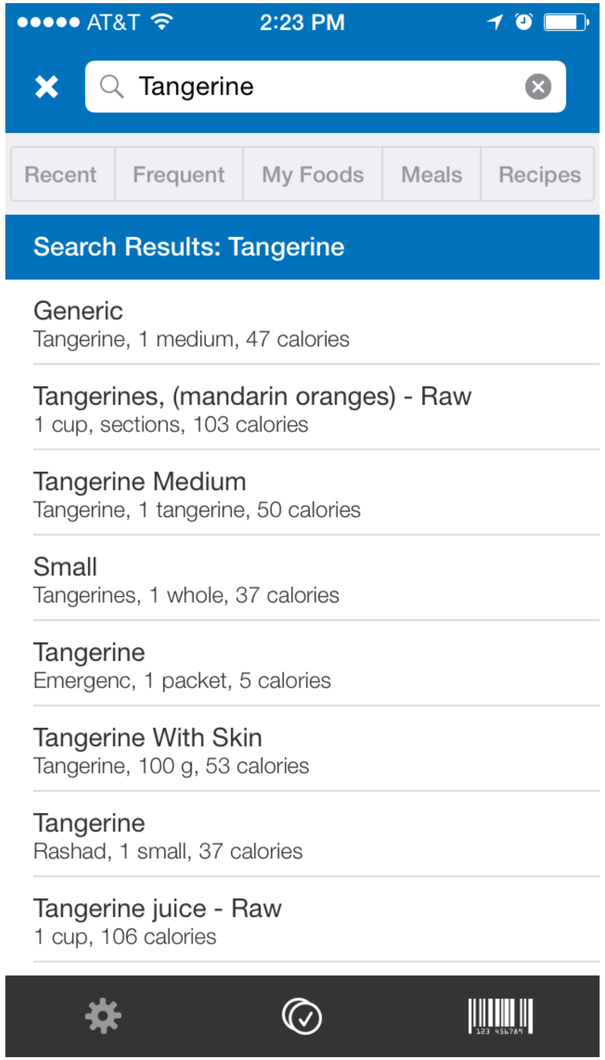
Aggregate the Nutritional Contents of the selected candidate, based on majority voting of the cluster members.



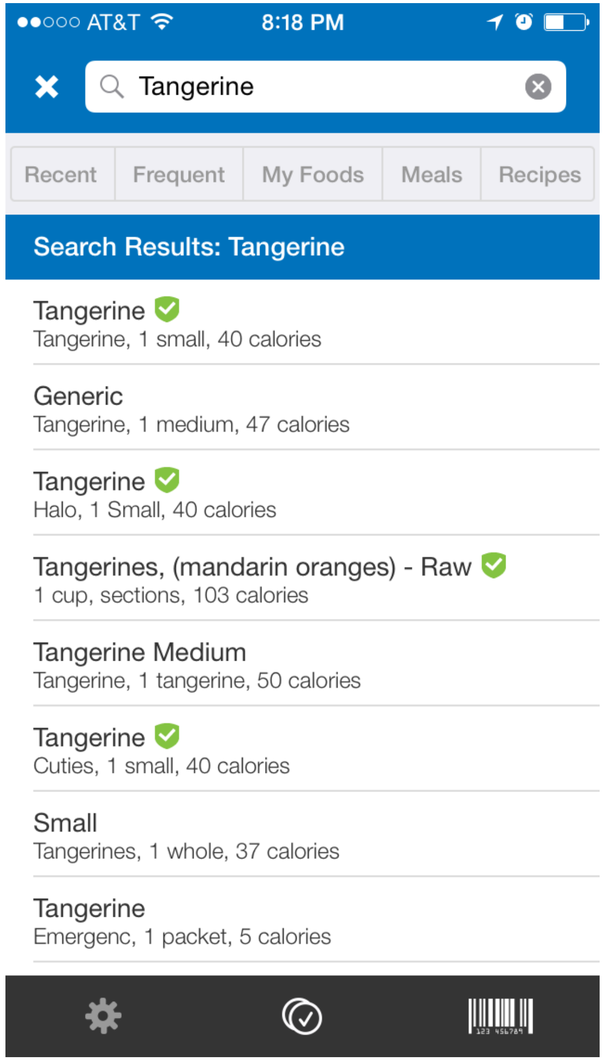
Independent for each cluster!

Result

Before



After



What Next?

- ☑ Improve the quality of verified foods based on user feedback from search

- ☑ Perform Link Analysis on the graph of users and foods nodes (e.g., PageRank)

Apache Spark GraphX Library!