Sentiment Analysis & Keyword Mining on Yelp Reviews

Fei Liu, Jing Song, Chen Wang, Dixin Yan

Project Goal

- Gain insights from Yelp review text via sentiment analysis
  - Can be used to classify positive/negative reviews
- Mine top keywords to describe a business
  - Can be used to explore different business attributes based on top keywords in the reviews

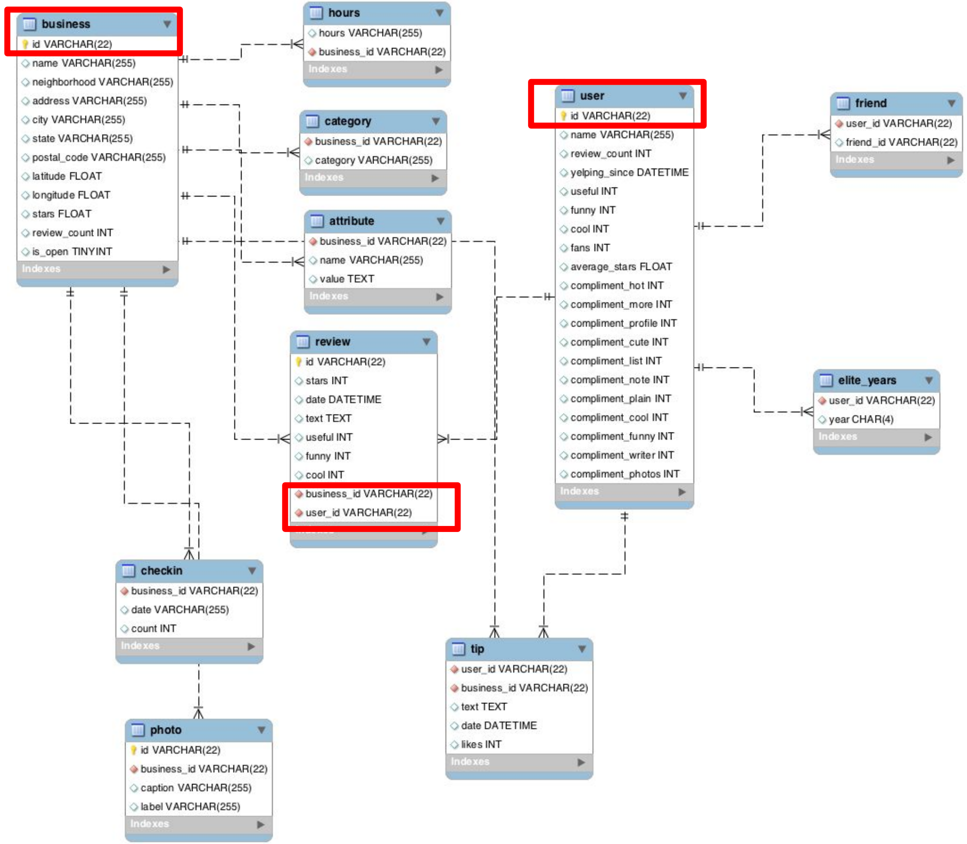
Data

- What's in a review? Is it positive or negative?


```
{
        "_id" : ObjectId("5a5d41a969b675a54a6f310a"),
        "review_id" : "VfBHSwC5Vz_pbFluy07i9Q",
        "user_id" : "cjpdDjZyprfyDG3RlkVG3w",
        "business_id" : "uYHaNptLzDLoV_JZ_MuzUA",
        "stars" : 5,
        "date" : "2016-07-12",
        "text" : "My girlfriend and I stayed here for 3 nights and loved it. The location of
this hotel and very decent price makes this an amazing deal. When you walk out the front door
 Scott Monument and Princes street are right in front of you, Edinburgh Castle and the Royal
Mile is a 2 minute walk via a close right around the corner, and there are so many hidden gem
s nearby including Calton Hill and the newly opened Arches that made this location incredible
.\n\nThe hotel itself was also very nice with a reasonably priced bar, very considerate staff
, and small but comfortable rooms with excellent bathrooms and showers. Only two minor compla
ints are no telephones in room for room service (not a huge deal for us) and no AC in the roo
m, but they have huge windows which can be fully opened. The staff were incredible though, le
tting us borrow umbrellas for the rain, giving us maps and directions, and also when we had l
ost our only UK adapter for charging our phones gave us a very fancy one for free.\n\nI would
 highly recommend this hotel to friends, and when I return to Edinburgh (which I most definit
ely will) I will be staying here without any hesitation.",
        "useful" : 0,
        "funny" : 0,
        "cool" : 0
} _
```

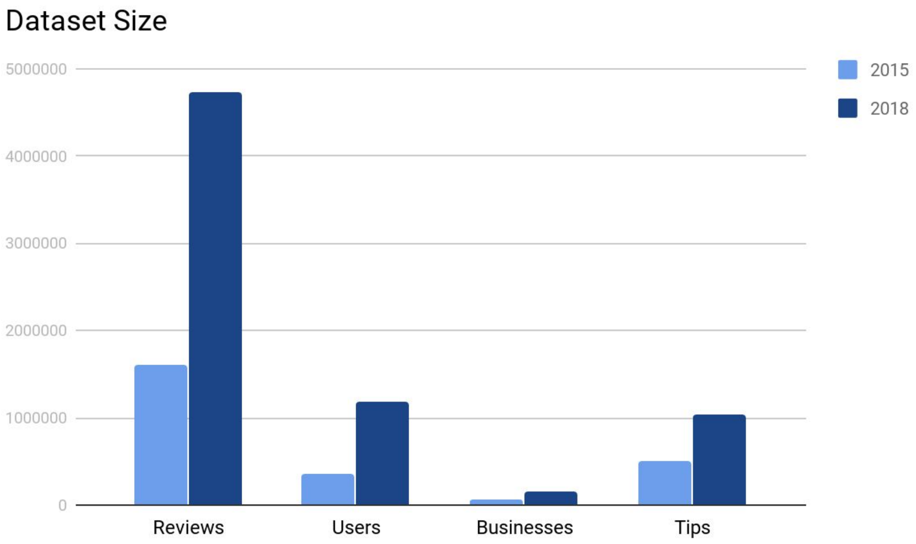
Data

- 5.79 GB data
- 4.7M reviews and 1M tips by 1.3M users for 157K businesses



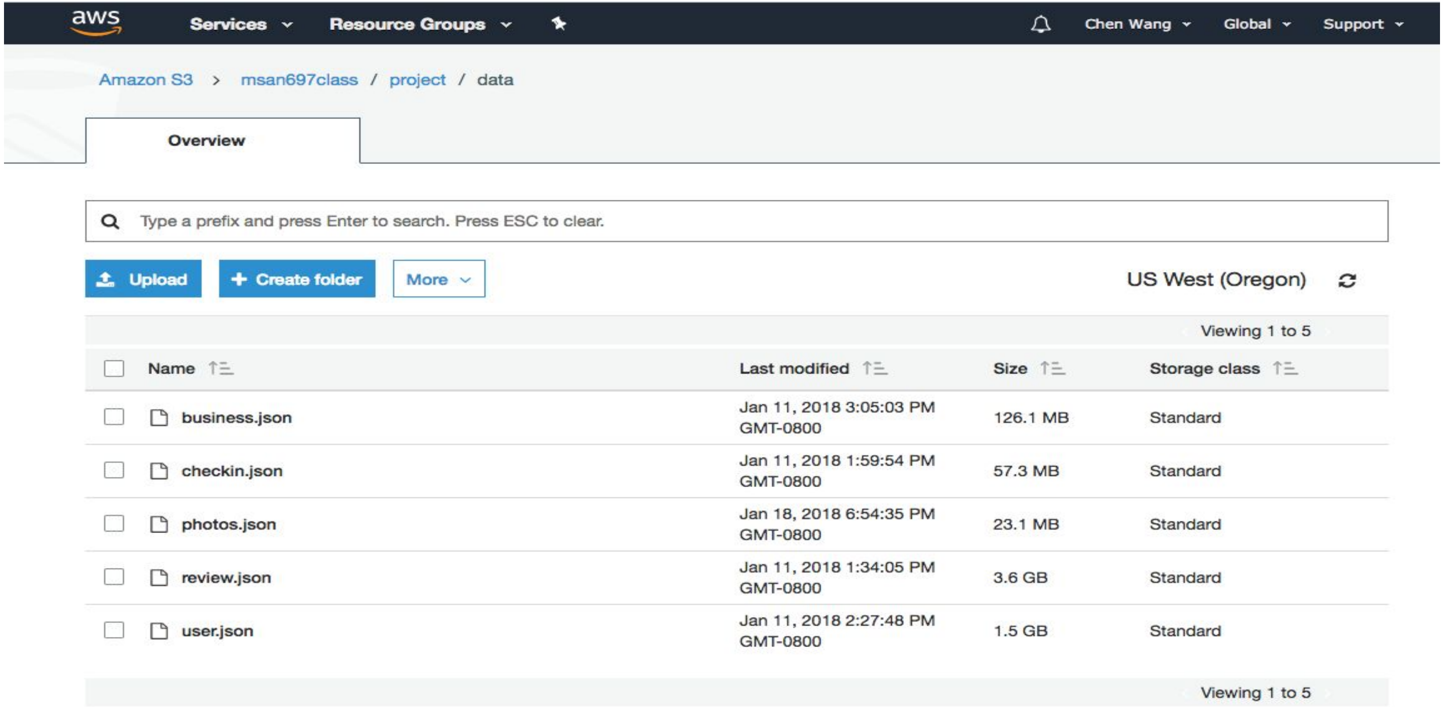
Data

- 5.79 GB data
- 4.7M reviews and 1M tips by 1.3M users for 157K businesses
- The size of the dataset has increased significantly in the past three years
    - requires scalable processing for large datasets



Pipeline

S3 Storage


aws
Services
Resource Groups
Chen Wang
Global
Support
Amazon S3 > msan697class / project / data
Overview
Type a prefix and press Enter to search. Press ESC to clear.
Upload
Create folder
More
US West (Oregon)
Viewing 1 to 5

Name	Last modified	Size	Storage class
business.json	Jan 11, 2018 3:05:03 PM GMT-0800	126.1 MB	Standard
checkin.json	Jan 11, 2018 1:59:54 PM GMT-0800	57.3 MB	Standard
photos.json	Jan 18, 2018 6:54:35 PM GMT-0800	23.1 MB	Standard
review.json	Jan 11, 2018 1:34:05 PM GMT-0800	3.6 GB	Standard
user.json	Jan 11, 2018 2:27:48 PM GMT-0800	1.5 GB	Standard

Viewing 1 to 5

Mongo Query

To show how many documents are there in each collection:

```
MSAN — ec2-user@ip-172-31-29-193:~ — ssh -i...
> use project
switched to db project
> db.business.count()
156639
> db.review.count()
4736897
> db.user.count()
1318510
>
```

EMR Instance Detail

Connections: --
Master public DNS: ec2-54-186-237-213.us-west-2.compute.amazonaws.com SSH
Tags: --

Model	vCPU	Mem (GiB)
m4.large	2	8

Summary

ID: j-1NWQ5DZ8DU4O4
Creation date: 2018-01-17 10:44 (UTC-8)
End date: 2018-01-17 17:57 (UTC-8)
Elapsed time: 7 hours, 13 minutes
Auto-terminate: No
Termination protection: Off

Configuration details

Release label: emr-5.11.0
Hadoop distribution: Amazon 2.7.3
Applications: Ganglia 3.7.2, Spark 2.2.1, Zeppelin 0.7.3
Log URI: s3://aws-logs-137120904662-us-west-2/elasticmapreduce/
EMRFS consistent view: Disabled
Custom AMI ID: --

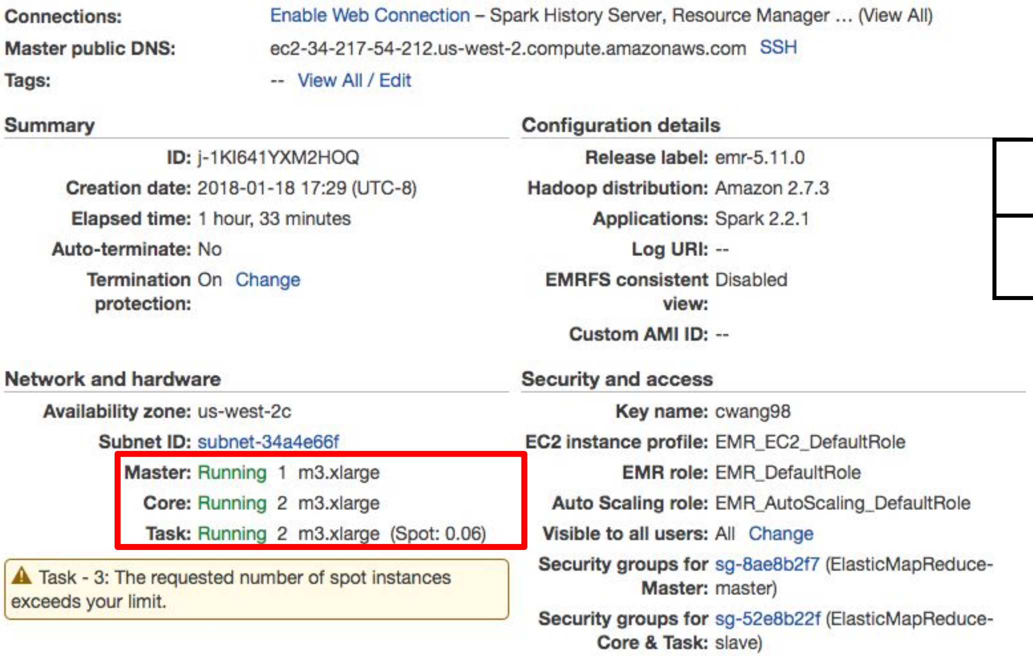
Network and hardware

Availability zone: us-west-2b
Subnet ID: subnet-9fe73ed7
Master: Terminated 1 m4.large
Core: Terminated 2 m4.large
Task: --

Security and access

Key name: msan694
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Visible to all users: All Change
Security groups for Master: sg-29fb2f55 (ElasticMapReduce-master)
Security groups for Core & Task: sg-9afa2ee6 (ElasticMapReduce-slave)

EMR Instance Detail



Connections: Enable Web Connection – Spark History Server, Resource Manager ... (View All)
Master public DNS: ec2-34-217-54-212.us-west-2.compute.amazonaws.com SSH
Tags: -- View All / Edit

Summary

ID: j-1KI641YXM2HOQ
Creation date: 2018-01-18 17:29 (UTC-8)
Elapsed time: 1 hour, 33 minutes
Auto-terminate: No
Termination protection: On Change

Configuration details

Release label: emr-5.11.0
Hadoop distribution: Amazon 2.7.3
Applications: Spark 2.2.1
Log URI: --
EMRFS consistent view: Disabled
Custom AMI ID: --

Network and hardware

Availability zone: us-west-2c
Subnet ID: subnet-34a4e66f
Master: Running 1 m3.xlarge
Core: Running 2 m3.xlarge
Task: Running 2 m3.xlarge (Spot: 0.06)

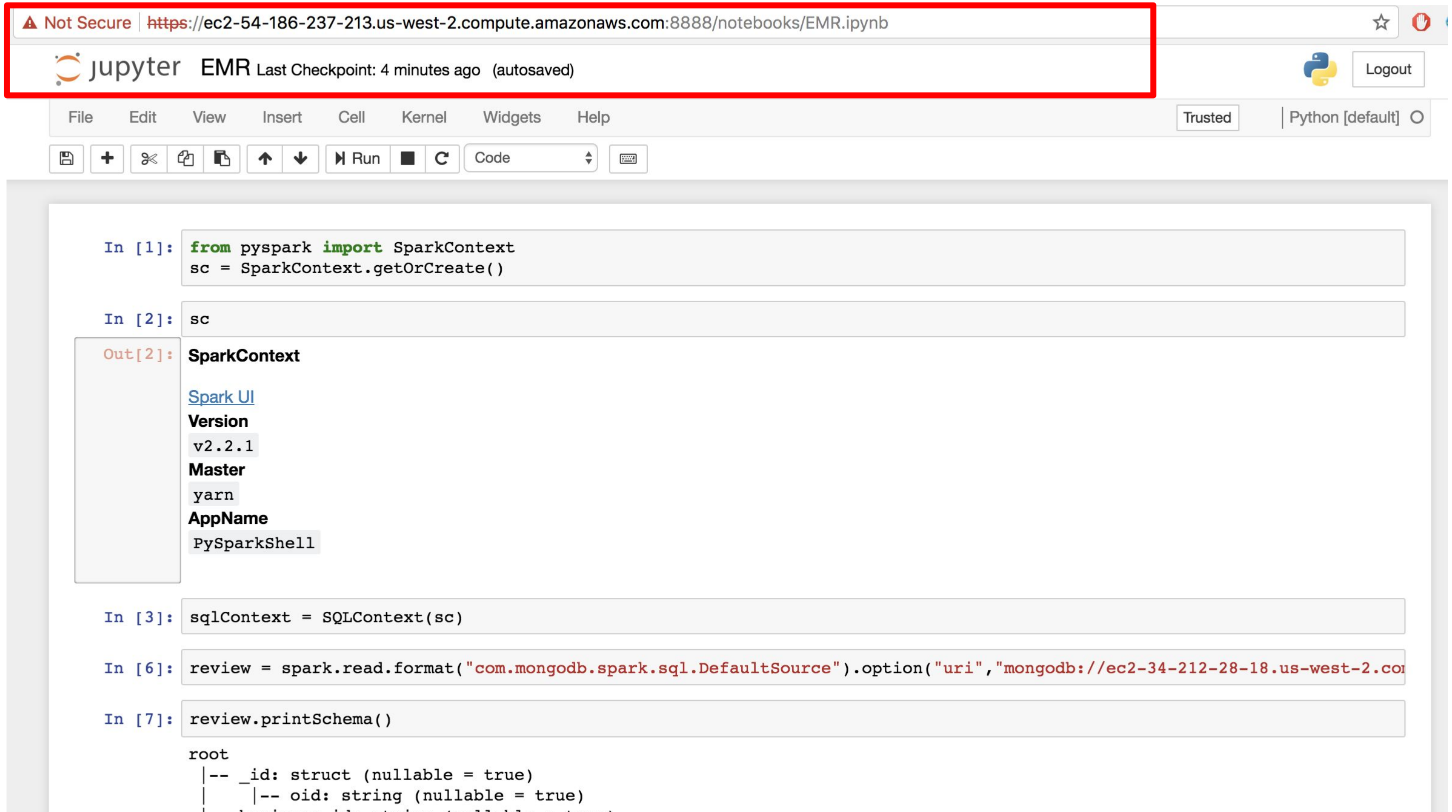
Task - 3: The requested number of spot instances exceeds your limit.

Security and access

Key name: cwang98
EC2 instance profile: EMR_EC2_DefaultRole
EMR role: EMR_DefaultRole
Auto Scaling role: EMR_AutoScaling_DefaultRole
Visible to all users: All Change
Security groups for Master: sg-8ae8b2f7 (ElasticMapReduce-master)
Security groups for Core & Task: sg-52e8b22f (ElasticMapReduce-slave)

Model	vCPU	Mem (GiB)
m3.xlarge	4	15

Running Spark on EMR



SparkSQL on EMR

ec2-34-217-53-92.us-west-2.compute.amazonaws.com

sparkml-yelp-d.c./ SparkSQL

jupyter SparkSQL Last Checkpoint: 7 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

In [1]:
```
from pyspark import SparkContext
sc = SparkContext.getOrCreate()
```

In [2]:
```
review_raw =spark.read.format("com.mongodb.spark.sql.DefaultSource").\
    option("uri","mongodb://ec2-54-202-247-153.us-west-2.compute.amazonaws.com/project.review").load()
biz_raw =spark.read.format("com.mongodb.spark.sql.DefaultSource").\
    option("uri","mongodb://ec2-54-202-247-153.us-west-2.compute.amazonaws.com/project.business").load()
```

In [9]:
```
biz_raw.filter("review_count >50").sort('stars',ascending = False). \
        select('name','review_count','city','stars').show(truncate=False)
```

```
+-------------------------------------+------------+----------+-----+
|name                                 |review_count|city      |stars|
+-------------------------------------+------------+----------+-----+
|Broken Dagger Tattoo Parlor          |131         |Las Vegas |5.0  |
|Inkenstein Tattoo Company            |77          |Glendale  |5.0  |
|Green Corner Restaurant              |267         |Mesa      |5.0  |
|Express Collision Center             |54          |Las Vegas |5.0  |
|Smallcakes Cupcakery and Creamery    |65          |Scottsdale|5.0  |
|Puff & Fluff Grooming and Pet Sitting|186         |Phoenix   |5.0  |
|A Master Mover                       |69          |Phoenix   |5.0  |
|David Frank Salon                    |68          |Scottsdale|5.0  |
|Tempe Eyecare Associates             |58          |Tempe     |5.0  |
|Bella Smiles                         |109         |Las Vegas |5.0  |
|Power Window Repair                  |91          |Phoenix   |5.0  |
|Affordable Technology Repairs        |60          |Las Vegas |5.0  |
|Arizona Livescan Fingerprinting      |54          |Phoenix   |5.0  |
|Hanedan                              |57          |Edinburgh |5.0  |
|Amelia C & Co                        |188         |Las Vegas |5.0  |
|Majestic Eyebrow Threading           |83          |Las Vegas |5.0  |
|Nails By Males                       |54          |Tempe     |5.0  |
|Coffee Talk                          |72          |Chandler  |5.0  |
|Pinkys Barbershop                    |59          |Phoenix   |5.0  |
|High Class Hookah Shop               |51          |Las Vegas |5.0  |
+-------------------------------------+------------+----------+-----+
only showing top 20 rows
```

Sentiment Analysis - Spark.ml

Input: Tfidf score extracted from Yelp review text

Output: positive (1) 

or negative (0) 

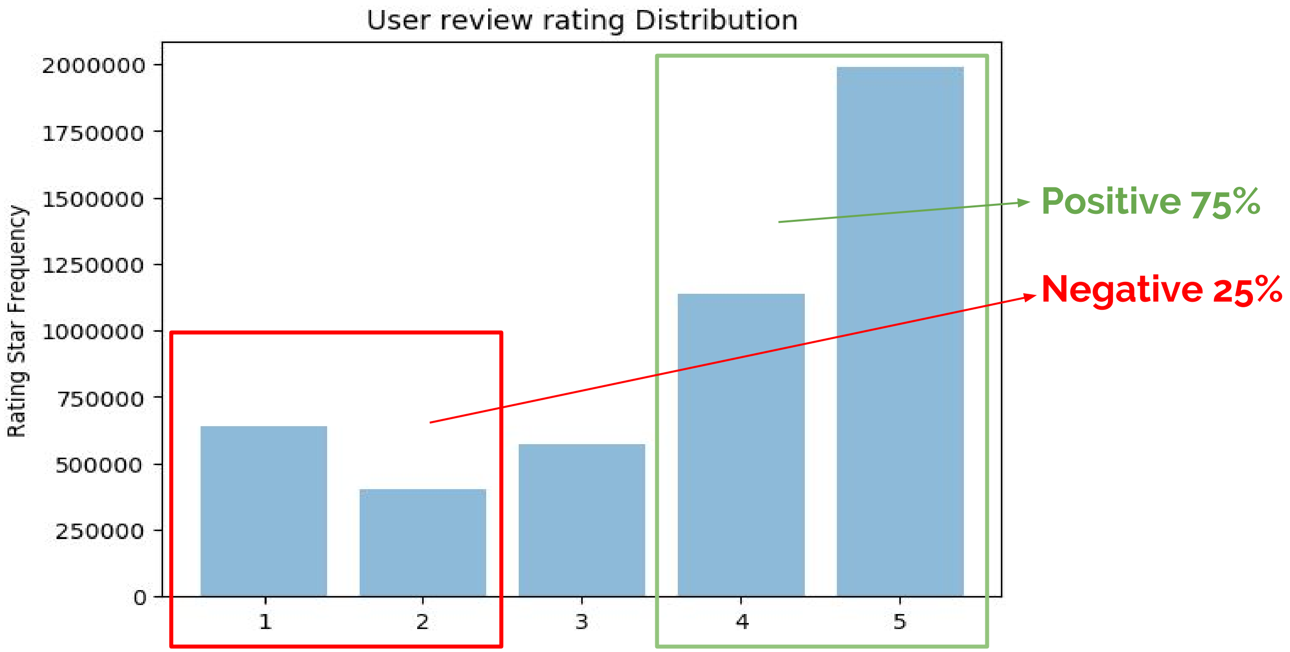
Data processing:

- Exclude reviews with 3 stars
- Collapse reviews with 4 and 5 stars as positive , 1 and 2 as negative
- Tokenize and remove stop words and punctuations from review text
- Unigram only for the purpose of sentiment analysis

Algorithm: Logistic Regression, Naive Bayes, Multilayer Perceptron Classifier

Glimpse of Data - Review Distribution

Trying to see if the issue of imbalanced data is present:



TFIDF Pipeline

Unigram tfidf

In [14]:
```
tokenizer = Tokenizer().setInputCol("text").setOutputCol("words")
remover= StopWordsRemover().setInputCol("words").setOutputCol("filtered").setCaseSensitive(False)
hashingTF = HashingTF().setNumFeatures(n_features).setInputCol("filtered").setOutputCol("rawFeatures")
idf = IDF().setInputCol("rawFeatures").setOutputCol("features").setMinDocFreq(0)
pipeline=Pipeline(stages=[tokenizer,remover,hashingTF,idf])
print "Tokenizer:"
print tokenizer.explainParams()
print "***************************"
print "Remover:"
print remover.explainParams()
print "***************************"
print "HashingTF:"
print hashingTF.explainParams()
print "***************************"
print "IDF:"
print idf.explainParams()
```

```
Tokenizer:
inputCol: input column name. (current: text)
outputCol: output column name. (default: Tokenizer_43c48e8a65d542f7e152__output, current: words)
***************************
Remover:
caseSensitive: whether to do a case sensitive comparison over the stop words (default: False, current: False)
inputCol: input column name. (current: words)
outputCol: output column name. (default: StopWordsRemover_4cd69a6ac67a1c68f913__output, current: filtered)
stopWords: The words to be filtered out (default: [u'i', u'me', u'my', u'myself', u'we', u'our', u'ours', u'ourselve
s', u'you', u'your', u'yours', u'yourself', u'yourselves', u'he', u'him', u'his', u'himself', u'she', u'her', u'her
s', u'herself', u'it', u'its', u'itself', u'they', u'them', u'their', u'theirs', u'themselves', u'what', u'which',
u'who', u'whom', u'this', u'that', u'these', u'those', u'am', u'is', u'are', u'was', u'were', u'be', u'been', u'bein
g', u'have', u'has', u'had', u'having', u'do', u'does', u'did', u'doing', u'a', u'an', u'the', u'and', u'but', u'if',
u'or', u'because', u'as', u'until', u'while', u'of', u'at', u'by', u'for', u'with', u'about', u'against', u'between',
u'into', u'through', u'during', u'before', u'after', u'above', u'below', u'to', u'from', u'up', u'down', u'in', u'ou
t', u'on', u'off', u'over', u'under', u'again', u'further', u'then', u'once', u'here', u'there', u'when', u'where',
u'why', u'how', u'all', u'any', u'both', u'each', u'few', u'more', u'most', u'other', u'some', u'such', u'no', u'no
r', u'not', u'only', u'own', u'same', u'so', u'than', u'too', u'very', u's', u't', u'can', u'will', u'just', u'don',
u'should', u'now', u"i'll", u"you'll", u"he'll", u"she'll", u"we'll", u"they'll", u"i'd", u"you'd", u"he'd", u"sh
e'd", u"we'd", u"they'd", u"i'm", u"you're", u"he's", u"she's", u"it's", u"we're", u"they're", u"i've", u"we've", u"y
ou've", u"they've", u"isn't", u"aren't", u"wasn't", u"weren't", u"haven't", u"hasn't", u"hadn't", u"don't", u"does
n't", u"didn't", u"won't", u"wouldn't", u"shan't", u"shouldn't", u"mustn't", u"can't", u"couldn't", u'cannot', u'coul
d', u"here's", u"how's", u"let's", u'ought', u"that's", u"there's", u"what's", u"when's", u"where's", u"who's", u"wh
y's", u'would'])
***************************
HashingTF:
binary: If True, all non zero counts are set to 1. This is useful for discrete probabilistic models that model binary
```

Model 1 Logistic regression

```
%%time
from pyspark.ml.classification import LogisticRegression
lr =  LogisticRegression(maxIter=100, regParam=0.01, elasticNetParam=0.8)
pipeline=Pipeline(stages=[tokenizer,remover,hashingTF,idf, lr])
logreg_model=pipeline.fit(train_set)
predictions = logreg_model.transform(test_set)
evaluate_metric(predictions)
```

Model 2 Naive Bayes

s://ec2-54-186-237-213.us-west-2.compute.amazonaws.com:8888/notebooks/sparkml-yelp

msan697_final_ML Last Checkpoint: an hour ago (autosaved)

View Insert Cell Kernel Widgets Help

Run Code

Model 2: Unigram Naive Bayes

```
%%time
nb = NaiveBayes(smoothing = 1.0, modelType = "multinomial")
pipeline=Pipeline(stages=[tokenizer,remover,hashingTF,idf, nb])
nb_model=pipeline.fit(train_set)
nb_prediction = nb_model.transform(test_set)
#print evaluation metrics
evaluate_metric(nb_prediction)
```

Model 3 Multilayer Perceptron Classifier

```
from pyspark.ml.classification import MultilayerPerceptronClassifier
layers = [n_features, 5 , 2]
trainer = MultilayerPerceptronClassifier(maxIter=10, layers=layers, blockSize=128, seed=1234)
pipeline=Pipeline(stages=[tokenizer,remover,hashingTF,idf, trainer])
nn_model = pipeline.fit(train_set)

nn_prediction = nn_model.transform(test_set)

evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction",
                                              metricName="f1")
f1 = evaluator.evaluate(nn_prediction)
print("F1_score = %0.4f" %(f1))

evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction",
                                              metricName="accuracy")
accuracy = evaluator.evaluate(nn_prediction)
print("Accuracy = %0.4f" %(accuracy))
```

Performance Evaluation

Define evaluation metrics

```
# compute accuracy on the test set
def evaluate_metric(predictions):

    evaluator = BinaryClassificationEvaluator().setMetricName("areaUnderROC")
    print 'Area under ROC curve:',evaluator.evaluate(predictions)

    evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction",
                                                  metricName="f1")
    f1 = evaluator.evaluate(predictions)
    print("F1_score = %0.4f" %(f1))

    evaluator = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction",
                                                  metricName="accuracy")
    accuracy = evaluator.evaluate(predictions)
    print "Accuracy = %0.4f" %(accuracy))
```

EMR Runtime & Metrics Comparison

	Logistic regression		Unigram Naive Bayes		Multilayer Perceptron Classifier	
	4 nodes m3.xlarge	2 nodes m4.large	4 nodes m3.xlarge	2 nodes m4.large	4 nodes m3.xlarge	2 nodes m4.large
Time	10 min	19min	5min	11min	8 min	23min
AUC	0.91		0.61		N/A	
F1_score	0.83		0.85		0.89	
Accuracy	0.85		0.85		0.89	

Keywords Mining from Business Reviews

- Questions
  - Which words differentiate positive reviews from negative ones?
  - What features/attributes are people mentioning in their reviews?

- Approach
  - Remove punctuation, numbers, stop words
  - Produce N-grams (Uni, Bi, Tri-grams)
  - Extract top frequent words

Keywords Mining - Example Code

```
def remove_num_punct(text):
    text = text.lower()
    my_string = text.replace("-", " ")
    regex = re.compile('[' + re.escape(string.punctuation) + '0-9\\r\\t\\n]')
    nopunct = regex.sub(" ", my_string)  # delete stuff but leave at least a space to avoid clumping together
    return nopunct

udf_num_punct = udf(lambda x:remove_num_punct(x))
```

```
def bi_gram(words):
    words = [w for w in words if len(w) > 0]
    bigram = [" ".join([words[i],words[i+1]])for i in range(len(words)-1)]
    return bigram
```

```
def tri_gram(words):
    words = [w for w in words if len(w) > 0]
    trigram = [" ".join([words[i],words[i+1],words[i+2]])for i in range(len(words)-2)]
    return trigram
```

```
tokenizer = Tokenizer(inputCol="text", outputCol="words")
remover = StopWordsRemover(inputCol="words", outputCol="filtered", caseSensitive=False)
pipeline=Pipeline(stages=[tokenizer,remover])
```

Functions to remove punctuations & numbers, and produce Bi-gram & Tri-gram

Tokenizer and StopWordsRemover Pipeline

Keywords Mining - Example Code

Top 15 N-gram from Positive Reviews

```
review_pos = review_pos.select(udf_num_punct('text').alias('text'))
pos_words=pipeline.fit(review_pos).transform(review_pos).select("filtered")
pos_rdd= pos_words.rdd.map(list).map(lambda x:x[0]).cache()
```

Convert dataframe to RDD to perform Map-Reduce functions

```
#Unigram
pos_uni=pos_rdd.flatMap(lambda words: [w for w in words if len(w) > 0])
top_pos_uni = pos_uni.map(lambda x: (x,1)).reduceByKey(lambda x,y: x+y).sortBy(lambda x: x[1], ascending=False)
#Bigram
pos_bi= pos_rdd.flatMap(lambda x: bi_gram(x))
top_pos_bi = pos_bi.map(lambda x: (x,1)).reduceByKey(lambda x,y: x+y).sortBy(lambda x: x[1], ascending=False)
#Trigram
pos_tri= pos_rdd.flatMap(lambda x: tri_gram(x))
top_pos_tri = pos_tri.map(lambda x: (x,1)).reduceByKey(lambda x,y: x+y).sortBy(lambda x: x[1], ascending=False)
```

Top 15 Words in Positive vs Negative Reviews

Unigram		Bigram		Trigram	
Positive	**Negative**	**Positive**	**Negative**	**Positive**	**Negative**
great	**food**	highly recommend	**customer service**	definitely come back	never go back
good	**place**	**first time**	**go back**	great customer service	worst customer service
place	**get**	**customer service**	**first time**	definitely go back	go somewhere else
food	**like**	really good	**come back**	wait go back	poor customer service
service	**service**	**go back**	even though	definetely coming back	horrible customer service
time	**one**	**come back**	didn't even	highly recommend placd	long story short
like	**back**	great place	tasted like	great food great	waste time money
one	**time**	great service	**came back**	best I've ever	never going back
get	**good**	**las vegas**	**going back**	food great service	won't going back
really	us	ice cream	front desk	mac n cheese	never come back
go	**go**	next time	much better	service great food	give zero stars
back	even	love place	told us	next time I'm	took minutes get
also	said	great food	**have ever**	back next time	worst service ever
best	never	**have ever**	last time	love love love	never coming back
have	didn't	every time	never go	definitely going back	asked speak manager

Single Business Analysis

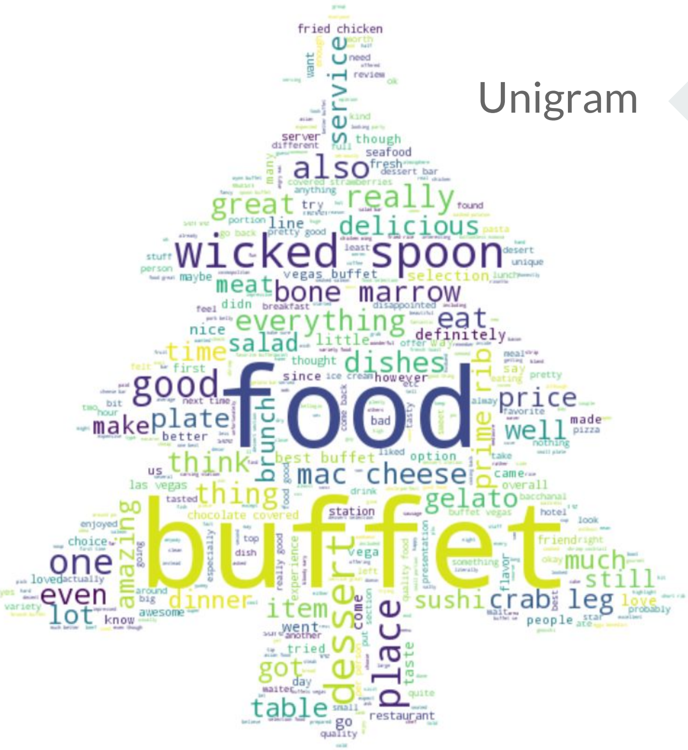
```python
# select the restaurants with the most number of reviews, seems that they are all from LV - high class buffet
biz_raw.sort('review_count',ascending = False).select('name','review_count','city').show(truncate=False)
```

name	review_count	city
Mon Ami Gabi	6979	Las Vegas
Bacchanal Buffet	6417	Las Vegas
Wicked Spoon	5632	Las Vegas
Gordon Ramsay BurGR	5429	Las Vegas
Earl of Sandwich	4789	Las Vegas
Hash House A Go Go	4371	Las Vegas
Serendipity 3	3913	Las Vegas
The Buffet	3873	Las Vegas
Lotus of Siam	3838	Las Vegas
The Buffet at Bellagio	3700	Las Vegas
ARIA Resort & Casino	3634	Las Vegas
The Cosmopolitan of Las Vegas	3621	Las Vegas
Secret Pizza	3542	Las Vegas
Bouchon at the Venezia Tower	3439	Las Vegas
Luxor Hotel and Casino Las Vegas	3429	Las Vegas
MGM Grand Hotel	3285	Las Vegas
Gangnam Asian BBQ Dining	3180	Las Vegas
McCarran International Airport	3090	Las Vegas
Hash House A Go Go	2963	Las Vegas
The Venetian Las Vegas	2951	Las Vegas

only showing top 20 rows

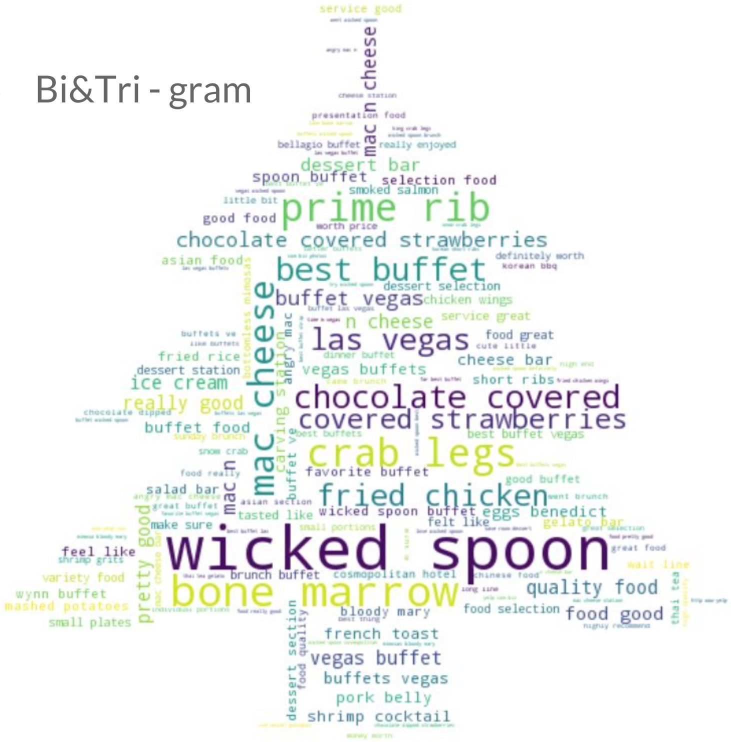
- Business with the most number of reviews - all from Las Vegas
- Use ‘Wicked Spoon’ as an example

Wicked Spoon’s Word Cloud



Unigram

Bi&Tri - gram




Love

Lessons learned

1. Tune hyperparameters
2. Limitation of running jobs on Jupyter Notebook on EMR requires packages to be installed across master and worker nodes
3. .collect() issue: collect dataframe failed on local and EMR; Solution is adding more memory to the driver or use other method instead
4. Request larger EMR if necessary; use spot pricing for lower price
5. Building the whole database pipeline takes a lot of efforts - better take notes along the way

Not tuning hyperparameters right

```
1  test_predicts.show()
```

```
|            features|label|       rawPrediction|         probability|prediction|
+--------------------+-----+--------------------+--------------------+----------+
|(262144,[14,78,61...|  0.0|[-0.6631775220879...|[0.34002618691923...|       1.0|
|(262144,[14,78,70...|  1.0|[-0.6631775220879...|[0.34002618691923...|       1.0|
|(262144,[14,78,76...|  1.0|[-0.6631775220879...|[0.34002618691923...|       1.0|
|(262144,[14,78,24...|  1.0|[-0.6631775220879...|[0.34002618691923...|       1.0|
|(262144,[14,329,9...|  1.0|[-0.6631775220879...|[0.34002618691923...|       1.0|
|(262144,[14,590,5...|  1.0|[-0.6631775220879...|[0.34002618691923...|       1.0|
|(262144,[14,614,1...|  1.0|[-0.6631775220879...|[0.34002618691923...|       1.0|
|(262144,[14,622,1...|  0.0|[-0.6631775220879...|[0.34002618691923...|       1.0|
|(262144,[14,991,2...|  1.0|[-0.6631775220879...|[0.34002618691923...|       1.0|
|(262144,[14,1133,...|  1.0|[-0.6631775220879...|[0.34002618691923...|       1.0|
|(262144,[14,1176,...|  1.0|[-0.6631775220879...|[0.34002618691923...|       1.0|
|(262144,[14,1846,...|  0.0|[-0.6631775220879...|[0.34002618691923...|       1.0|
|(262144,[14,1846,...|  0.0|[-0.6631775220879...|[0.34002618691923...|       1.0|
|(262144,[14,1846,...|  1.0|[-0.6631775220879...|[0.34002618691923...|       1.0|
|(262144,[14,1846,...|  1.0|[-0.6631775220879...|[0.34002618691923...|       1.0|
|(262144,[14,1889,...|  0.0|[-0.6631775220879...|[0.34002618691923...|       1.0|
|(262144,[14,1998,...|  1.0|[-0.6631775220879...|[0.34002618691923...|       1.0|
```

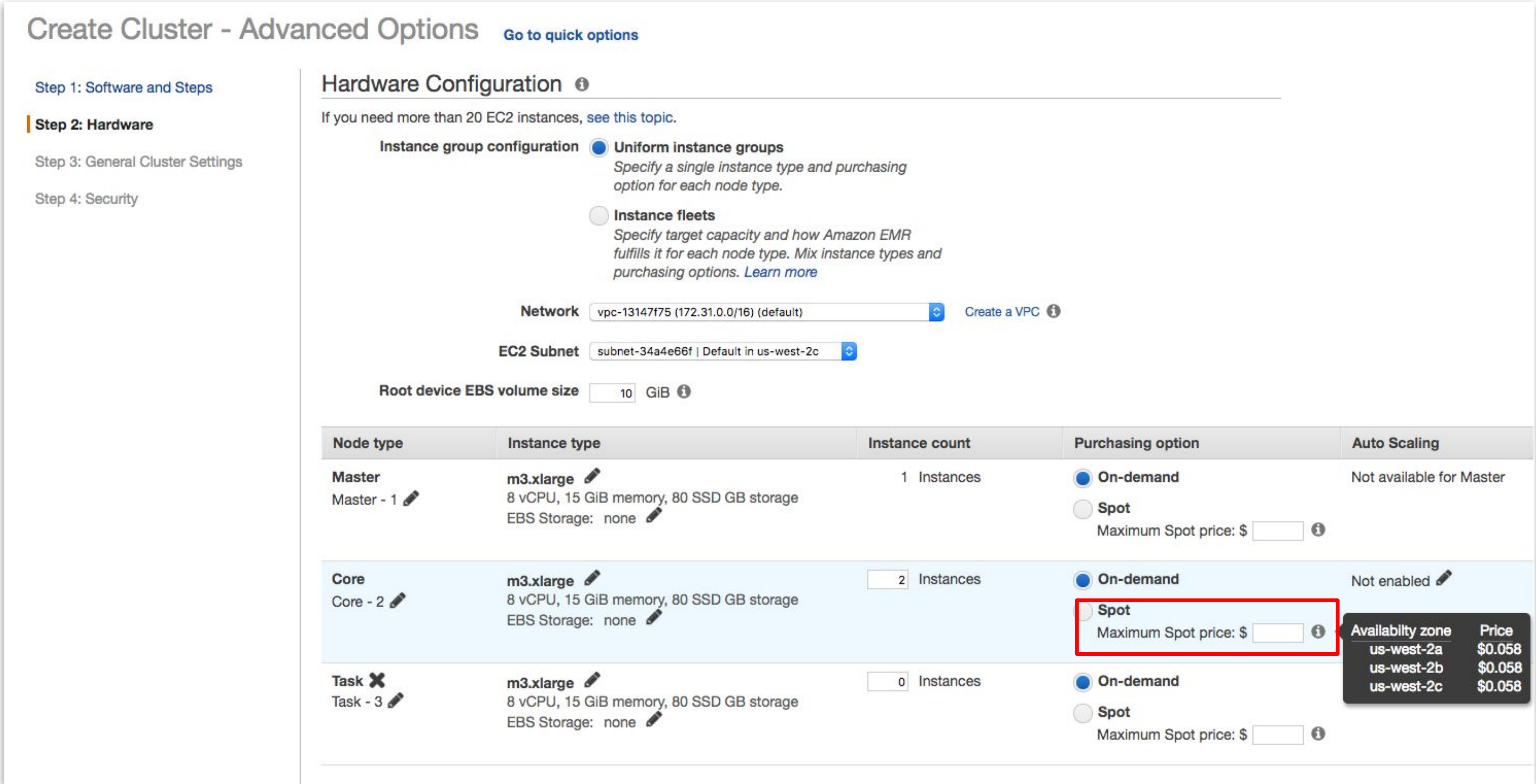
```
1  test_predicts.select(F.sum(test_predicts.predic
```

```
+---------------+
|sum(prediction)|
+---------------+
|        43090.0|
+---------------+
```

```
1  test_predicts.select(F.sum(test_predicts.label
```

```
+----------+
|sum(label)|
+----------+
|   28563.0|
+----------+
```

Use spot instances & bid for spot price



Thank you!


Welalin
Terma Kasih
Kiitos
Salamat
Merci
Grazie
Dankon
Multumesc
Spasibo
Asante
Thank
Matondo
Nirringrazzjak
Matur Nuwun
Ua Tsaug Rau Koj
Dank Je
Raibh Maith Agat
Merci
Mochchakkeram
Obrigado
You
Arigato
Salamat
Grazie
Kia Ora
Matondo
Welalin