Yelp Review Sentiment Analysis

Zihao Xu
Yuxuan Hong
James Ren

Agenda

- Problem & Motivation
- Data Description
- Exploratory Analysis
- Sentiment Analysis
- Future Directions

Problem & Motivation

- Explore interesting data trends/patterns
- Analyze sentiment changes over time
- Build a rating predictor based on review sentiment

Dataset Description

From Yelp Data Challenge

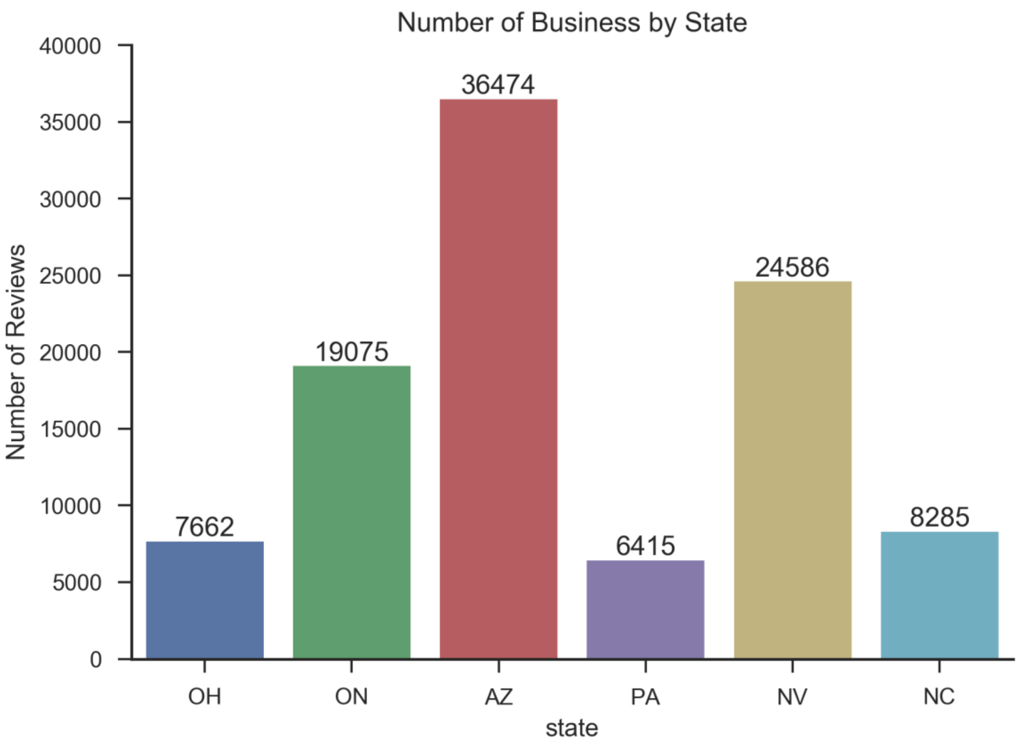
- Review.csv
- Business.csv
- User.csv

Dataset Description

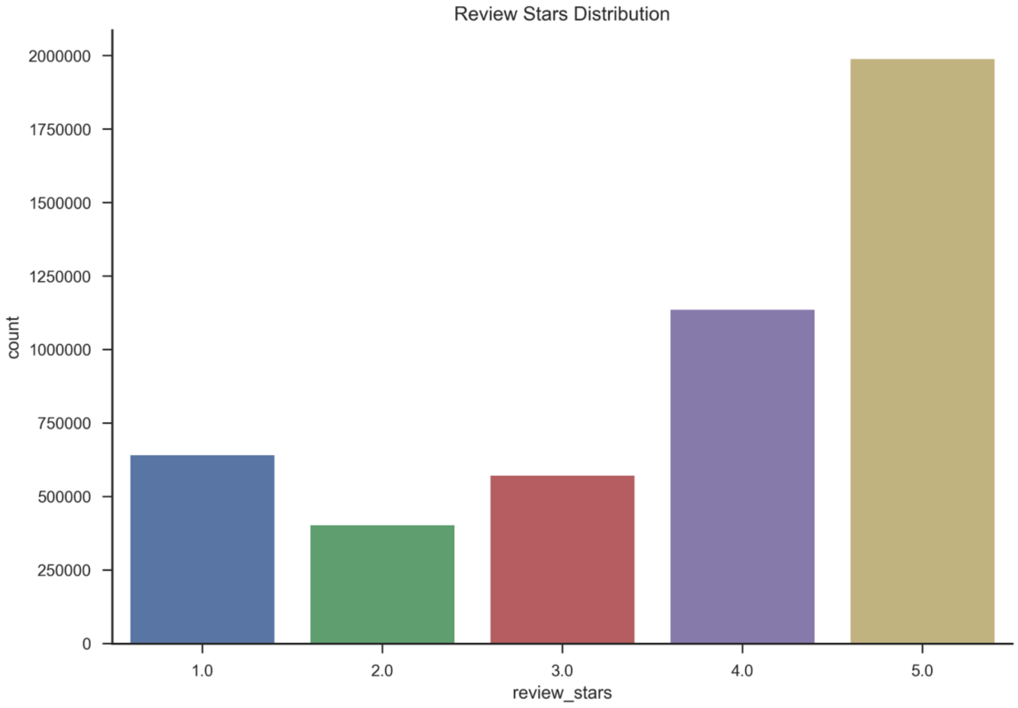
Facts:

- unique reviews:	4737000
- unique users:	970000
- unique businesses:	102500
- time span:	2004 - 2017
- size of *review.csv*:	3.9 GB

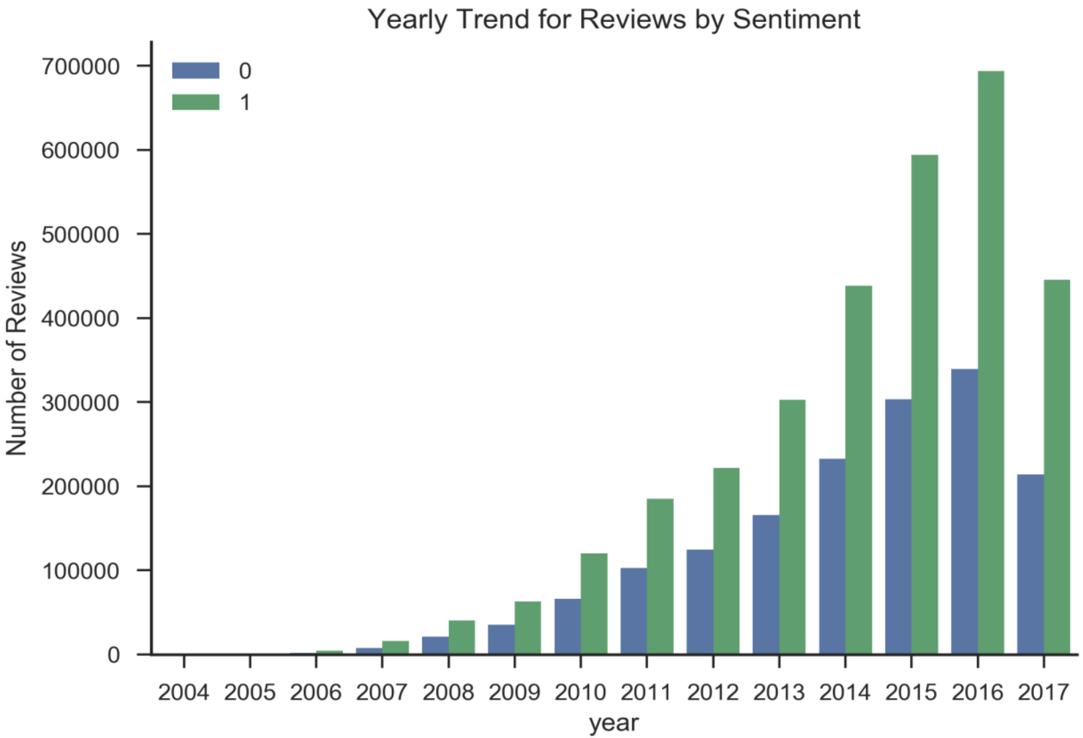
Dataset Description



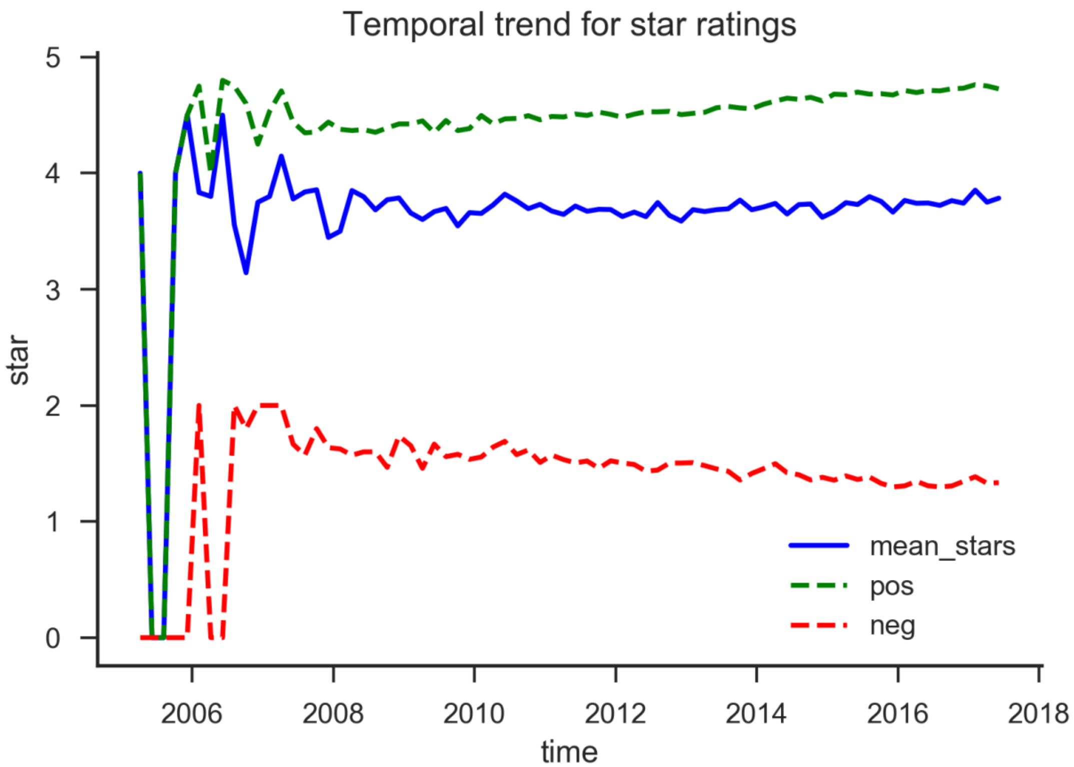
Dataset Description



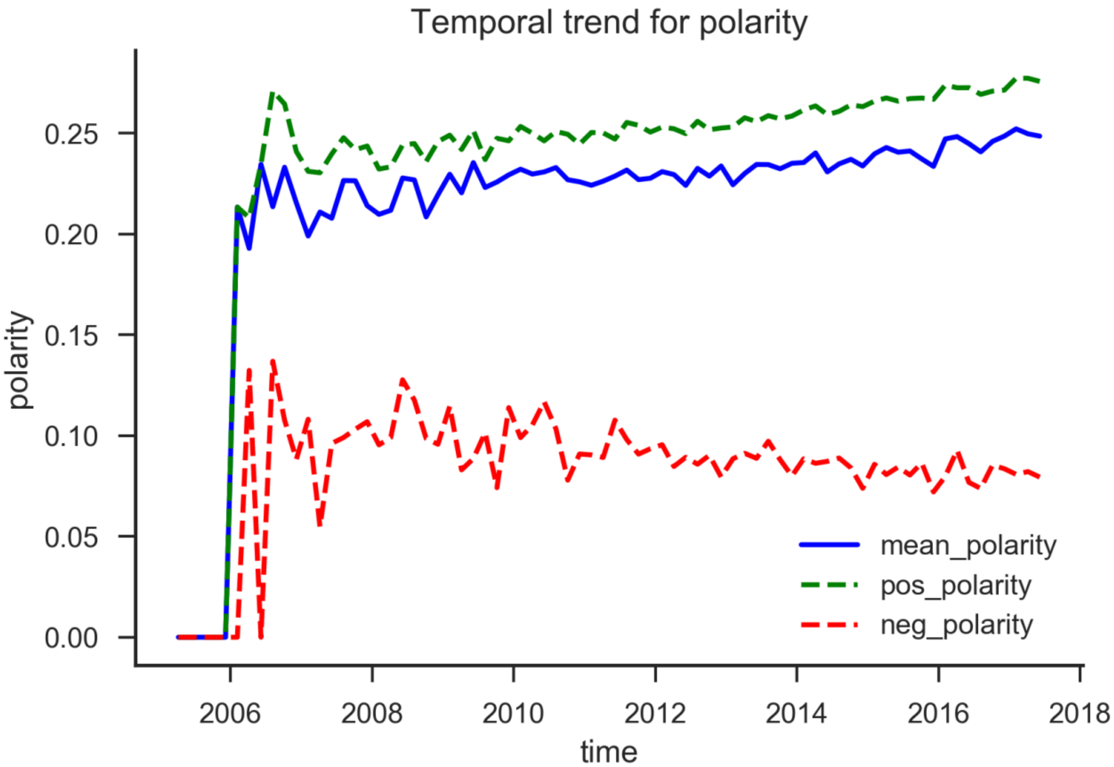
Dataset Description



Exploratory Analysis



Exploratory Analysis



Exploratory Analysis

Review Tags:

- Useful
- Funny
- Cool

8/15/2017

1 check-in

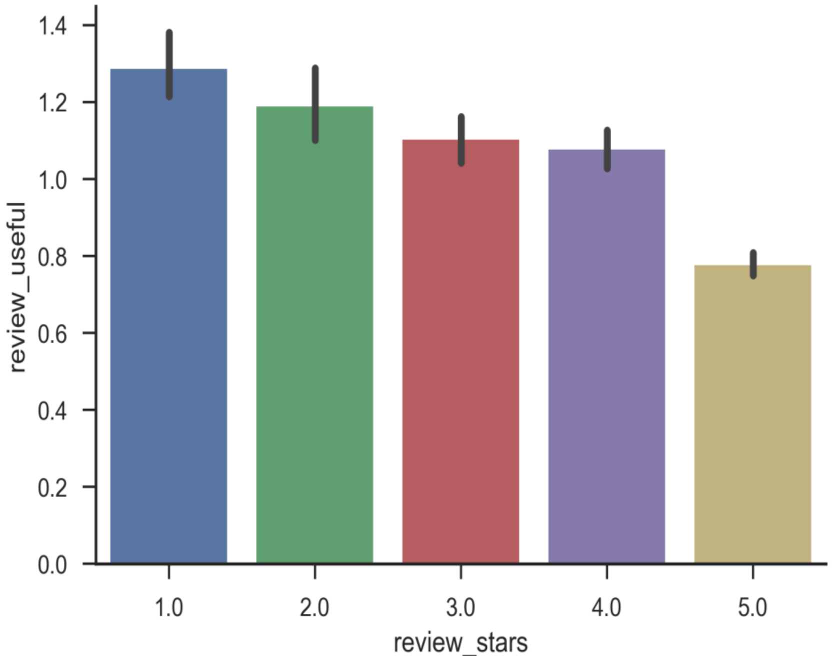
The menu was not very attractive to me because I don't eat much meat, but when I went into the restaurant, I found that the atmosphere was very good. It was clean, tidy, and had a traditional style. We felt so relaxed and happy to sit down and order. We had the beef kabob plate and the vegetarian sampler. Both were very delicious and the service was very very nice.

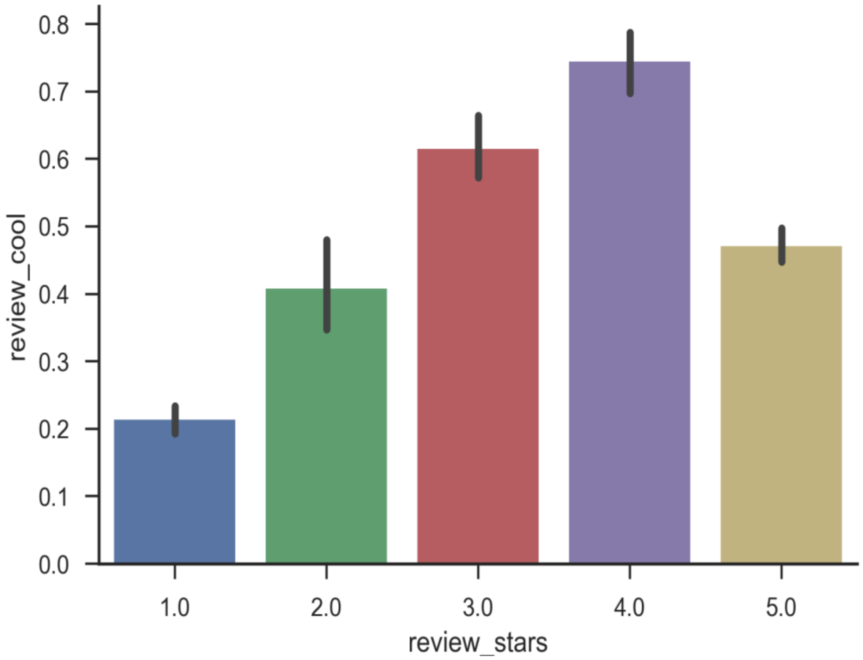
Enjoy!

Gary D. and 1 other voted for this review

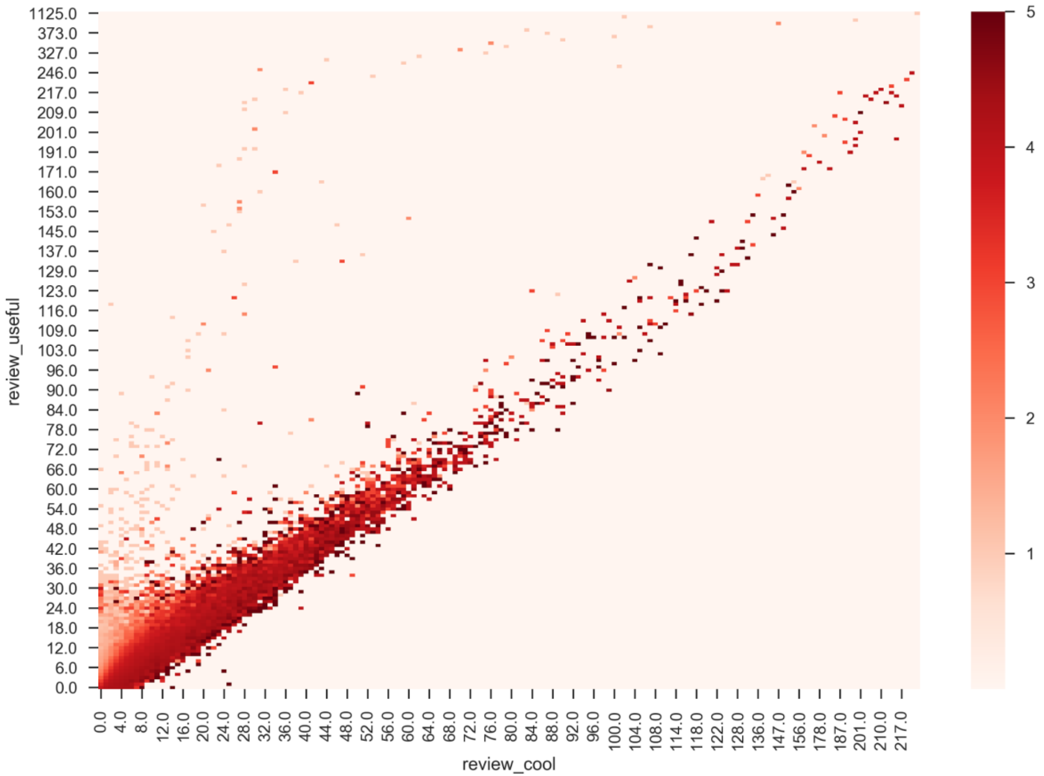
Useful 1 Funny 2 Cool

Exploratory Analysis

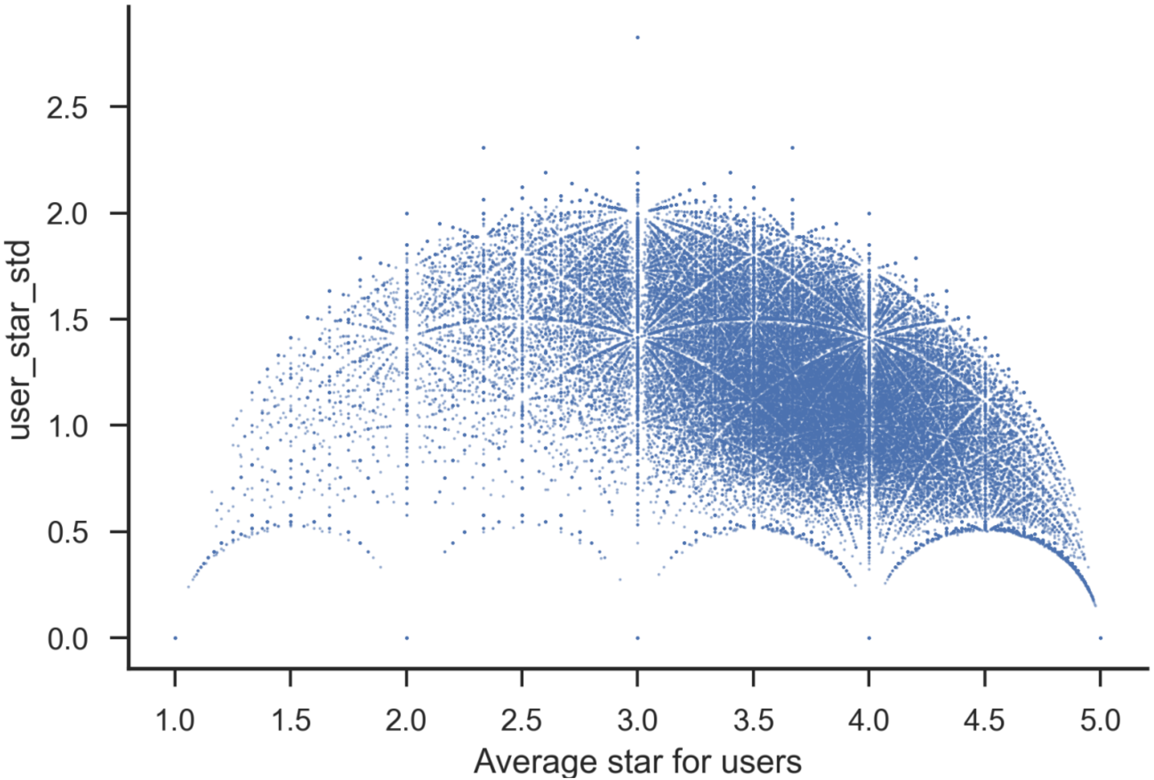




Exploratory Analysis



Mean vs. Standard Deviation



Sentiment Analysis

Methods:

- Select *[‘review_text’, ‘review_stars’]*
- Encode review_star: “pos”, “neg”
- Stem review_text and remove stop words
- Generate a sparse matrix representation of text
- ML models training and evaluation
- Ensemble methods: voting classifier

Sentiment Analysis

Text processing

Example = *"The python programmer named pythoner is pythoning a game pythonly"*

- **Word_tokenize**
  ['The', 'python', 'programmer', 'named', 'pythoner', 'is', 'pythoning', 'a', 'game', 'pythonly']
- **Clean_format**
  ['the', 'python', 'programmer', 'named', 'pythoner', 'is', 'pythoning', 'a', 'game', 'pythonly']
- **Remove_stop_words**
  ['the', 'python', 'programmer', 'named', 'pythoner', 'pythoning', 'game', 'pythonly']
- **PorterStemmer**
  ['the', 'python', 'programm', 'name', 'python', 'python', 'game', 'pythonli']

Sentiment Analysis

Raw Counts vs. Tfidf Weights:

,

['the', 'python', 'programm', 'name', 'python', 'python', 'game', 'pythonli']

Raw Counts:

[1, 1, 2, 1, 1, 3]

Tfidf Weights:

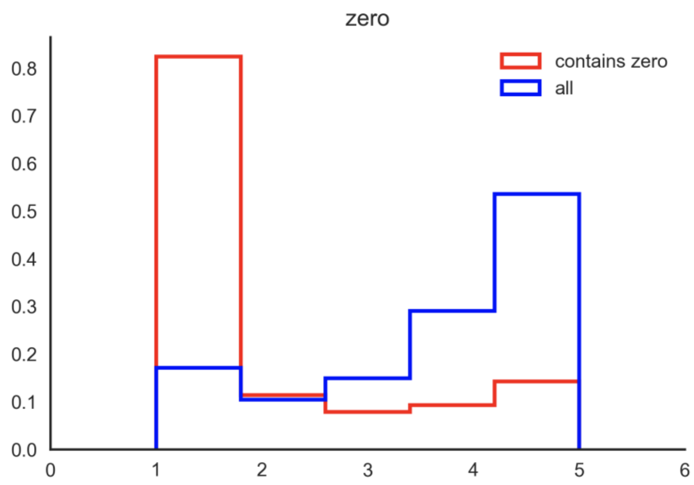
[0.102, 0.994, 0.045, 0.872, 0.453, 0.175, 0.032, 0.927, 0.321, 0.124, 0.142]

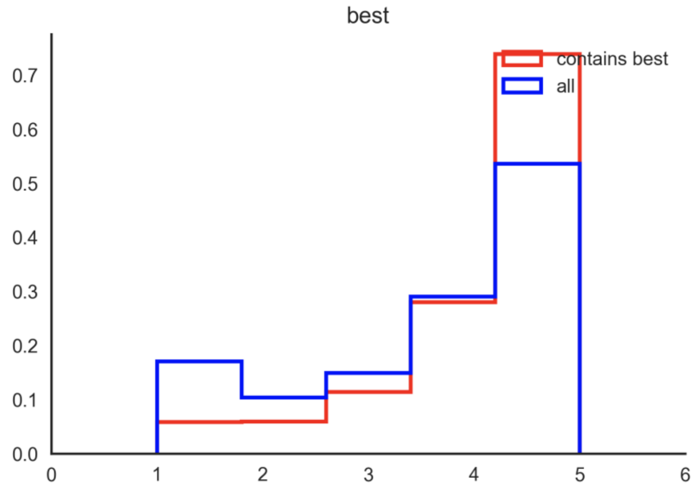
$$\text{tf-idf(t,d)} = \text{tf(t,d)} \times \text{idf(t)} \qquad \text{idf}(t) = log\frac{1+n_d}{1+\text{df}(d,t)} + 1$$

Sentiment Analysis

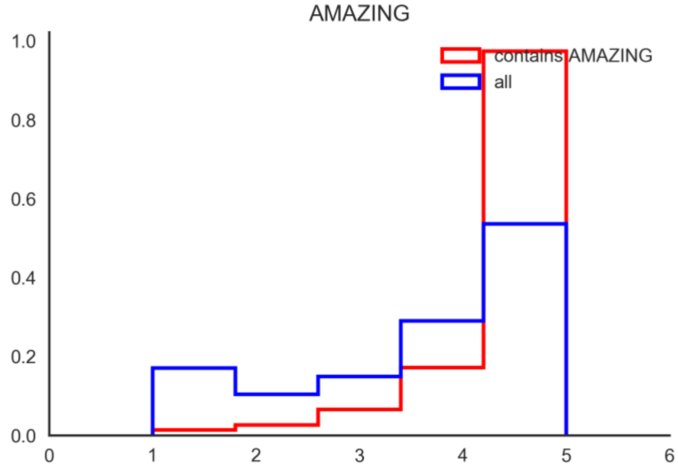
Features considered:

- Individual words:	“best”	“never”
- 2-grams:	“best place”	“never ever”
- Capitalized words:	“BEST”	“NEVER”


zero
contains zero
all
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0
1
2
3
4
5
6


best
contains best
all
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0
1
2
3
4
5
6


WORST
contains WORST
all
0.0
0.5
1.0
1.5
2.0
0
1
2
3
4
5
6


AMAZING
contains AMAZING
all
0.0
0.2
0.4
0.6
0.8
1.0
0
1
2
3
4
5
6

Model Training

Benchmark:	75% of all reviews are positive
Train / Test size:	140,917 / 46,972

Steps:

- Naive Bayes using Counts with only individual words
- More advanced clfs using Tfidf with only individual words
- More advanced clfs using Tfidf with words + 2-grams + CAPITAL
- Voting clf that combines top 3 clfs

First attempt: Naive Bayes

45% test accuracy… BUT!

```
Most Informative Features
        unprofession = 'unprofession'    neg : pos    =    125.2 : 1.0
            incompet = 'incompet'        neg : pos    =     92.1 : 1.0
             downhil = 'downhil'         neg : pos    =     61.2 : 1.0
            unaccept = 'unaccept'        neg : pos    =     53.0 : 1.0
               worst = 'worst'           neg : pos    =     47.1 : 1.0
           tasteless = 'tasteless'       neg : pos    =     45.1 : 1.0
         disrespect = 'disrespect'      neg : pos    =     42.0 : 1.0
                yuck = 'yuck'            neg : pos    =     40.6 : 1.0
                "no" = '"no"'            neg : pos    =     38.6 : 1.0
             really? = 'really?'         neg : pos    =     35.3 : 1.0
               shrug = 'shrug'           neg : pos    =     34.7 : 1.0
             disgust = 'disgust'         neg : pos    =     32.9 : 1.0
              aggrav = 'aggrav'          neg : pos    =     32.7 : 1.0
               appal = 'appal'           neg : pos    =     31.5 : 1.0
              ordeal = 'ordeal'          neg : pos    =     30.7 : 1.0
```

Importance of n-grams

Before using 2-grams

```
[[10321  1504]
 [ 1118 34030]]
Test accuracy:  0.944180699551
AUC:  0.920501743037
```

```
# The issue with n-grams: do not, not recommend, not good
print(clf.predict(vect.transform(['do not recommend this place',
                                  'this place is not good'])))
```

```
[1 1]
```

After using 2-grams

```
[[10820  1005]
 [  661 34487]]
Test accuracy:  0.964532816725
AUC:  0.94810218993
```

```
# NO MORE issue with n-grams: do not, not recommend, not good
print(clf.predict(vect.transform(['do not recommend this place',
                                  'this place is not good'])))
```

```
[0 0]
```

More Advanced Classifier...

Classifiers Tried:

- **8 Clfs:** MNB, BernoulliNB, LogisticRegression, LinearSVC, PolySVC, RadialSVC, NuSVC, SGDClassifier…

- **Best ones:**

	AUC_score	**Test accuracy**
LogisticRegression:	95.48%	95.35%
LinearSVC:	96.45%	94.81%
SGDClassifier:	94.71%	94.77%

More Advanced Classifier...

```
Smallest Coefs:
['worst' 'two star' 'not' 'disappoint' 'not worth' 'bland' 'terribl'
 'horribl' 'mediocr' 'veri disappoint' 'rude' 'meh' 'aw' 'at best' 'overpr'
 'poor' 'lack' 'not good' 'wors' 'not recommend' 'disgust' 'dirti' 'no'
 'never again' 'will never' 'no thank' 'wast' 'to love' 'wo be' 'noth'
 'not impress' 'will not' 'gross' 'poorli' 'not great' 'wo' 'unfortun'
 'underwhelm' 'elsewher' 'unprofession' 'suck' 'not veri' 'ruin'
 'never come' 'definit not']

Biggest Coefs:
['delici' 'great' 'amaz' 'awesom' 'excel' 'love' 'perfect' 'best'
 'not disappoint' 'fantast' 'good' 'be disappoint' 'definit' 'you wo'
 'highli recommend' 'outstand' 'perfectli' 'not bad' 'realli good' 'happi'
 'wonder' 'not too' 'friendli' 'never disappoint' 'thank' 'better than'
 'alway' 'love thi' 'my onli' 'four star' 'go wrong' 'the best'
 'will definit' 'not onli' 'fun' 'yummi' 'easi' 'tasti' 'so good'
 'profession' 'ca wait' 'recommend' 'fabul' 'love the' 'veri good']
```

More Advanced Classifier...

Let’s Try it out!

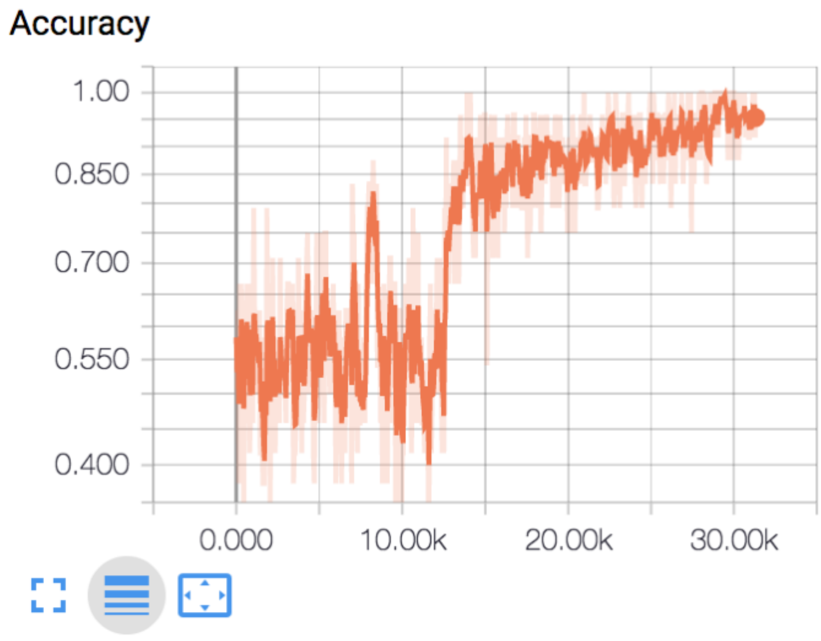
Future Directions

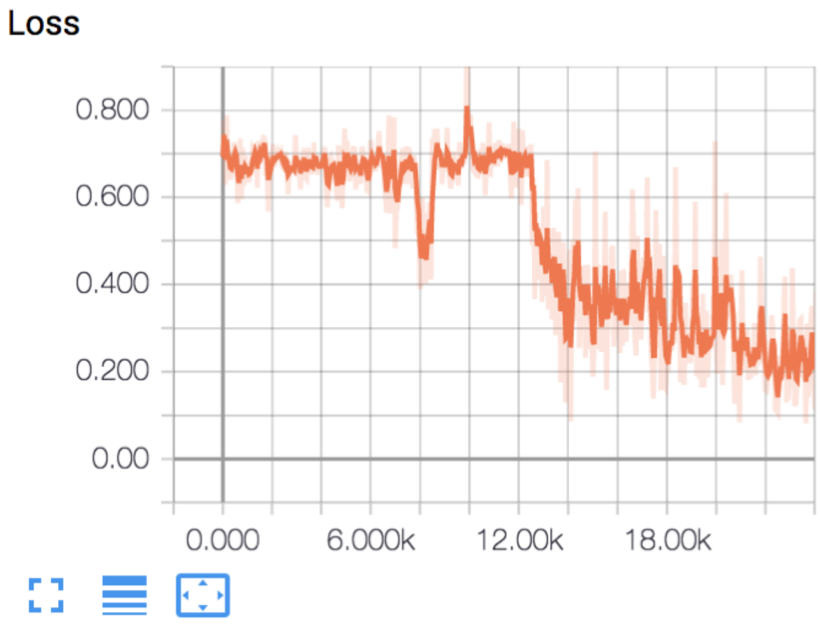
- Manually go over misclassified reviews
- Cross-validate the model to improve accuracy
- Predict the actual review_stars
- Try even more advanced models like LSTM

LSTM(Long-Short Term Memory network)

- RNN recurrent neural networks
- Capable of understanding context
- Example: I used to like this place when I was young, but not anymore.

Future Directions





Thank you for listening!