Unsupervised Sentiment Analysis of Yelp Reviews Using Natural Language Processing

Presented by Ben Chamblee
Springboard School of Data
6/15/21

Problem

- Online reviews are a common way for a customer to share their thoughts on a service/product
- With so much data it's impossible for humans to read every review
- Most websites have a '5 star' rating system - which can be inconsistent
    - My 4-star isn't the exact same as your 4-star and vice versa
- **The Solution?**
    - Create a machine learning model that determine sentiment based on the text within the review

Applying Sentiment Analysis to Yelp Reviews

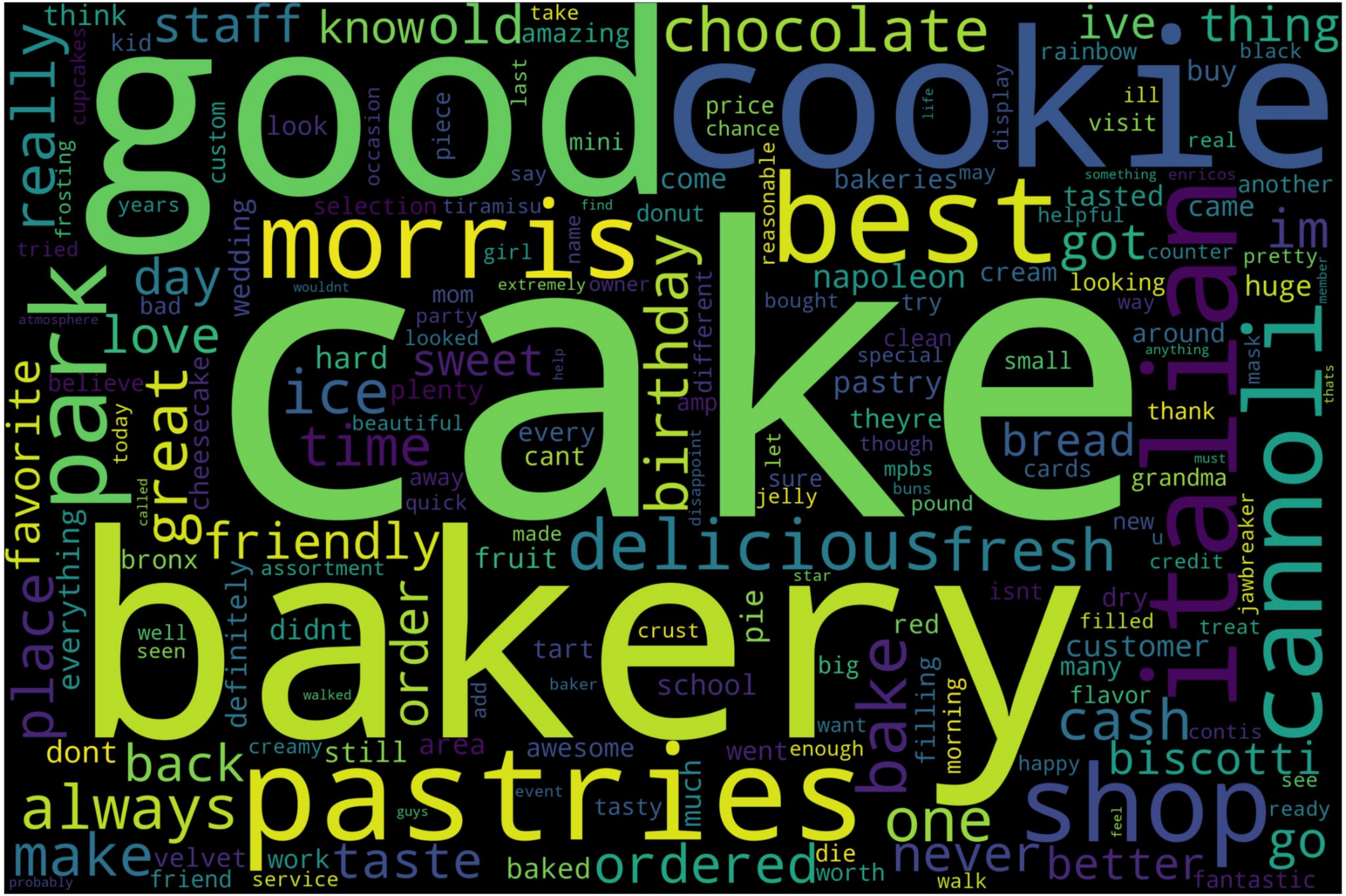
Data Wrangling

- Step 1: Use Yelp API to access specific URLs for each location
- Step 2: Use NYC Open Data to create a list of 200+ restaurants in New York (Populated, plenty of reviews)
- Step 3: Use python to scrape reviews from each restaurant’s webpage (robots.txt)
- Step 4: Make the reviews readable by filtering out html script using BeautifulSoup
- **End Result: 10,000+ unique reviews from 200+ restaurants**

```
▶<div class=" margin-b2__373c0__abANL border-color--default
373c0__2oFDT">…</div>
▼<div class=" margin-b2__373c0__abANL border-color--default
373c0__2oFDT">
  ▼<p class="comment__373c0__1M-px css-n6i4z7">
    ▶<span class=" raw__373c0__3rcx7" lang="en">…</span> ==
    </p>
  </div>
▶<div class=" margin-t3__373c0__1l90z margin-b2__373c0__abA
border-color--default__373c0__2oFDT">…</div>
▶<div class=" arrange__373c0__UHqhV vertical-align-middle__
3c0__2TQsQ border-color--default__373c0__2oFDT">…</div>
</div>
</li>
▶<li class=" margin-b5__373c0__2ErL8 border-color--default__37
__2oFDT">…</li>
▶<li class=" margin-b5__373c0__2ErL8 border-color--default__37
__2oFDT">…</li>
▶<li class=" margin-b5__373c0__2ErL8 border-color--default__37
__2oFDT">…</li>
▶<li class=" margin-b5__373c0__2ErL8 border-color--default__37
__2oFDT">…</li>
▶<li class=" margin-b5__373c0__2ErL8 border-color--default__37
```

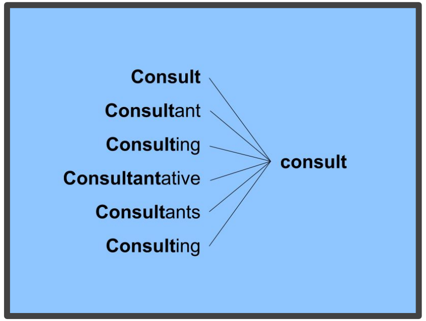
```
['Small location with limited seating, but 80s new
wave playing and friendly baristas. I dropped in fo
r an iced coffee after my workout at the NYSC next
door. Not much else to say. Pretty standard bucks m
enu. Got my drink quickly. Not much seating in this
one. Large communal table smack in the middle, and
some window seating all crowded up with wifi hogs a
t every seat.Basically a grab and go location to m
e. All good and A-OK.',
```

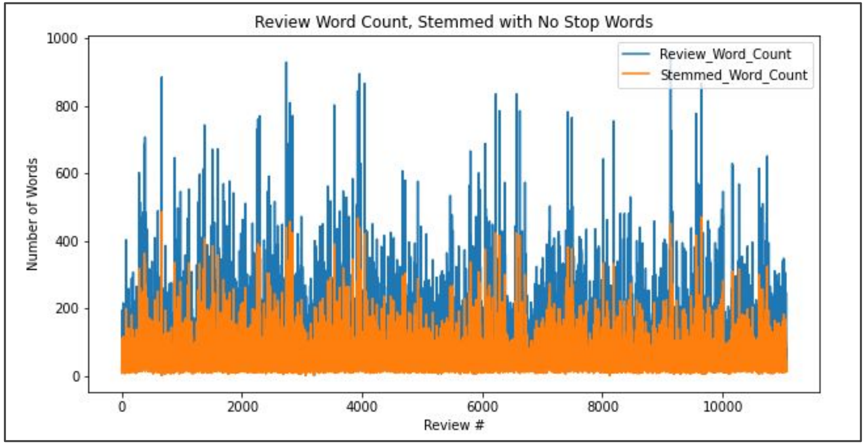
Word Clouds



cake
bakery
good
cookie
pastries
italian
cannoli
best
morris
park
shop
delicious
fresh
chocolate
staff
know
old
think
take
amazing
ive
thing
rainbow
black
buy
kid
really
cupcakes
last
price
chance
ill
visit
real
look
piece
custom
occasion
mini
life
display
frosting
say
come
bakeries
may
something
enricos
another
years
selection
tiramisu
find
donut
reasonable
tasted
helpful
came
im
tried
girl
name
got
counter
pretty
wouldnt
extremely
owner
napoleon
cream
looking
huge
member
day
wedding
mom
bought
try
way
bad
party
birthday
atmosphere
looked
different
clean
around
anything
love
hard
sweet
help
special
small
mask
thats
believe
plenty
pastry
favorite
great
ice
amp
theyre
thank
beautiful
every
time
bread
today
cheesecake
cant
disappoint
let
though
must
away
mpbs
cards
grandma
quick
sure
jelly
buns
pound
called
friendly
made
new
jawbreaker
everything
bronx
fruit
u
credit
assortment
star
place
definitely
order
pie
isnt
dry
crust
red
filled
treat
well
seen
didnt
tart
customer
big
bake
filling
many
morning
flavor
walked
baker
school
add
want
cash
contis
dont
back
creamy
still
area
awesome
went
enough
biscotti
happy
see
event
always
guys
tasty
much
one
ready
feel
make
velvet
work
taste
baked
ordered
die
worth
never
better
go
probably
friend
service
walk
fantastic

Text Stemming





- In order to save time and computing power, it's important to remove stop words (the, and, for, etc.)

- It's also important to implement stemming to make sure all word forms are conjugated the same

- Here you can see review word counts with (blue) and without stemming (orange)

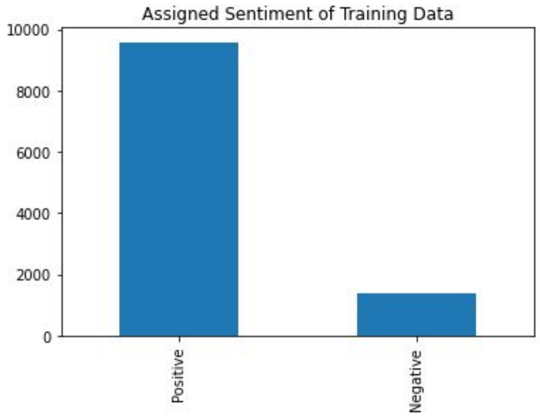
Classification

- Due to limit of requests when web scraping, the training data was unlabeled

- Had to assign sentiment using TextBlob's Polarity Value

- Uses Text2Vector to compare word vectors and find the most similar to known positive and negative word vectors

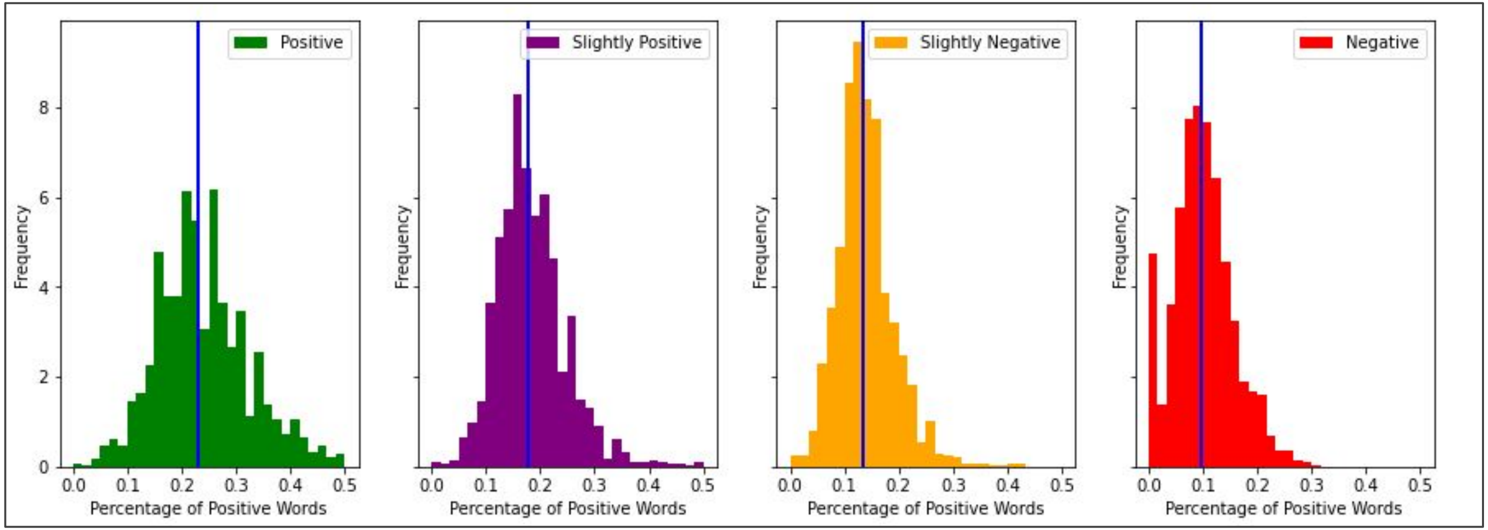
1st Split:



2nd Split:

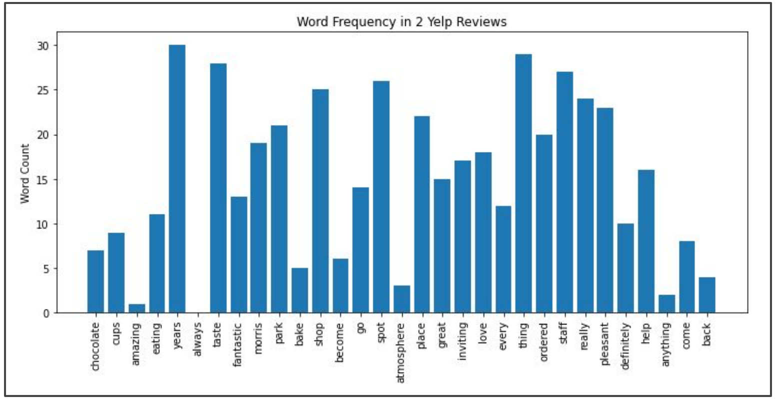
Feature Processing - Numerical

- Most features (stop word %, word count, stemmed words) didn't have a distinct difference between classifiers
- Percentage of Positive Words stood out as the average consistently decreases as you go from Positive to Negative

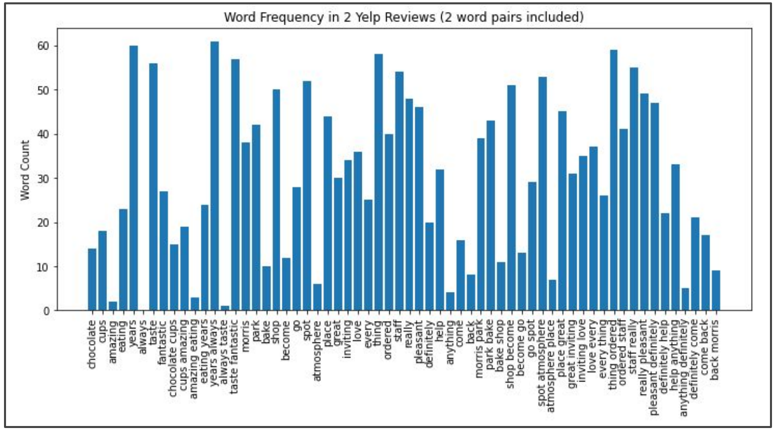


Feature Processing - Text

- **TFIDF:** Term Frequency Inverse Document Frequency takes a count of words for all 10,000 reviews and assigns a score based on how often they appear in one review

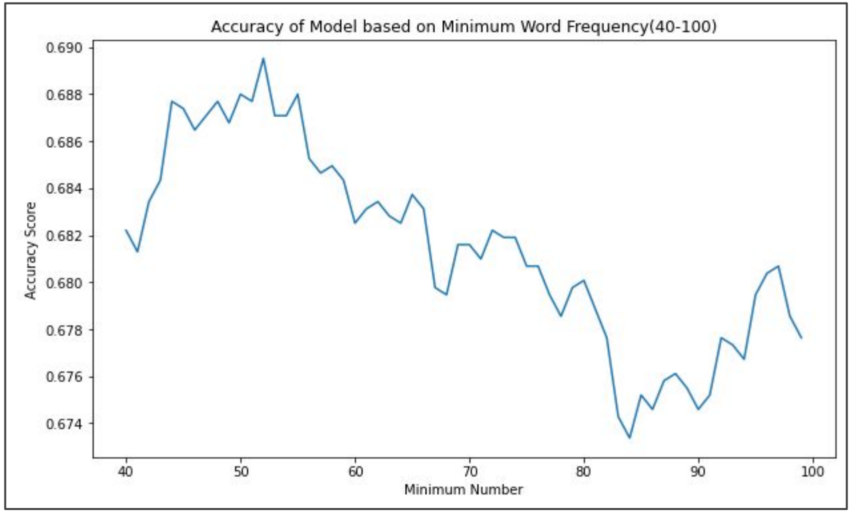


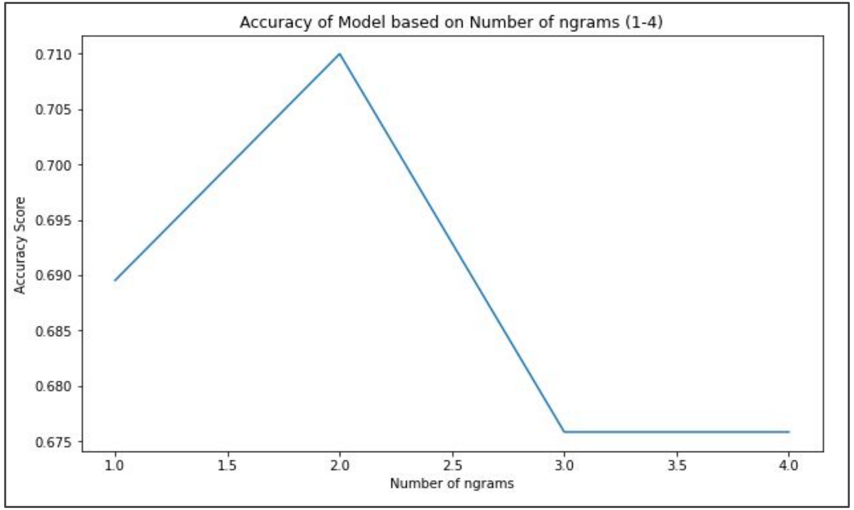
- **Ngrams:** Number of word token pairs



- How do we avoid overfitting by not having too many text features?

Feature Processing - Text





- Set minimum word frequency to 52 and number of ngrams to 2
- Boosted our baseline accuracy from 0.66 to 0.71
- Reduced the number of features from 9,000+ to around 1,600

Model Selection

- Tested 10 models in total
- **Part 1: ML With Extracted Features**
  - Logistic Regression
  - Random Forest
  - Support Vector Classification
  - Gradient Boosted Classification
- **Part 2: ML With Bag of Words**
  - Naive Bayes with CountVectorizer
  - Naive Bayes with TFIDF
  - Gradient Boosted with CountVectorizer
  - Gradient Boosted with TFIDF
- **Part 3: Ensemble Techniques**
  - Naive Bayes Probability Dense/Sparse
  - Stacked Model

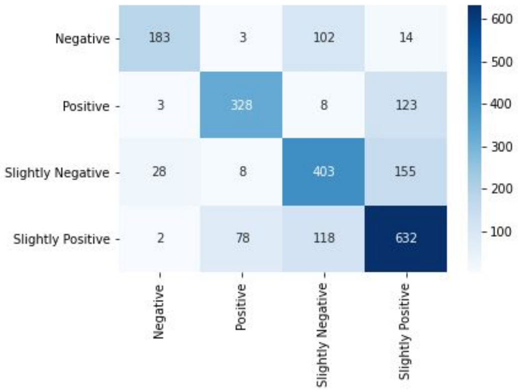
Model	Accuracy	F1_Macro	F1_Micro	F1_Weighted
Stacked Model	0.922	0.914	0.902	0.942
NB Probability Dense/Sparse	0.712	0.714	0.712	0.712
GBC	0.707	0.708	0.707	0.707
SVC	0.701	0.702	0.701	0.701
Logistic Regression	0.695	0.693	0.695	0.696
NB_TF	0.680	0.667	0.680	0.673
Random Forest	0.660	0.662	0.660	0.660
GBC_TF	0.629	0.628	0.629	0.627
GBC_CV	0.611	0.599	0.611	0.606
NB_CV	0.602	0.608	0.602	0.601

Stacked Model

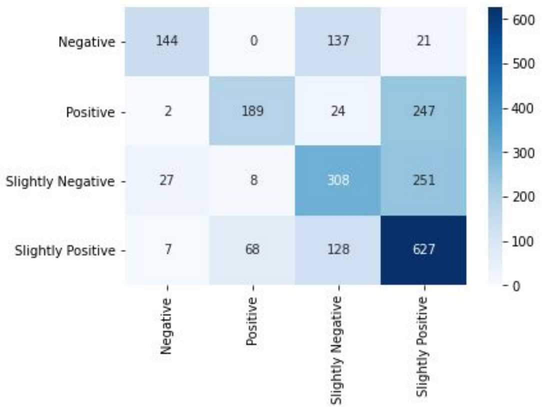
- Combination of two models:
  - Gradient Boosted Classification with TFIDF vectors and % of Positive Words as input
  - Naive Bayes with Dense/Sparse text matrix as input

- Use Naive Bayes Model to Calculate Probability of sparse matrix being either Positive, Negative, Slightly Positive, or Slightly Negative
  - Add this as a feature to our Gradient Boosted Classification

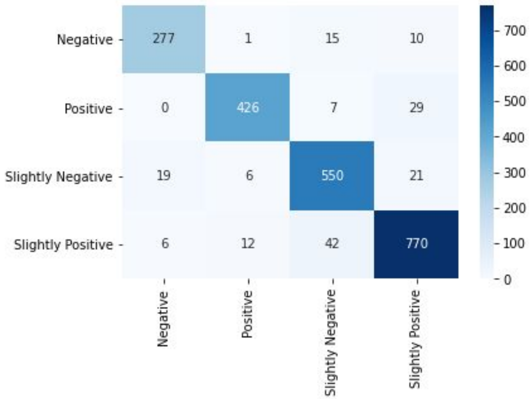
Gradient Boosted Classification



Naive Bayes



Stacked Model



Application

- This model, when deployed can determine sentiment analysis for yelp review with up to **92% Accuracy**
  - This will improve over time as more data is collected

- You can assign a ‘sentiment score’ to each location and find out which specific terms are making the score what it is
  - Would be very valuable for business owners to know how people feel but also what is making them feel that way

- Much more informative than the ‘5 star’ system

```
0.2361111111111111
Sentiment: Slightly Positive

scallop          0.57062
humid           0.282382
scooted         0.268943
triggered       0.257299
swilling        0.235943
mistaken         0.21589
renomy          0.214895
beginning       0.206109
discs           0.195823
2004            0.168366
horrified       0.158308
eclectically    0.154481
regions         0.152659
buffs           0.148592
rolatini        0.138212
```

Example from an Italian Restaurant:
- Looks like the guest enjoyed the scallops
- But the humidity might be why it’s just slightly positive
- A Compliment and something for the restaurant to work on

Conclusion/Findings

- You can really do anything with machine learning, as long as you have good data and the right model

- Web scraping is challenging, Yelp reviews are 40% stop words on average

- NLP has many applications, this is just scratching the surface of what’s possible

- Multi Classification is more informative, but much more difficult to increase accuracy

Thank you!

Questions?