Text Mining on Yelp Reviews

Mengjie Wang

Content

- Introduction
- Data Preprocessing
- Feature Extraction
  1. Bag of N-grams
  2. TF - IDF
- Results
- Other Try
- Future Work

Introduction

Dataset: Yelp Dataset Challenge (Round 11)

business, checkin, review, tip, user

The Challenge

We challenge students to use our data in innovative ways and break ground in research. Here are some examples of topics we find interesting, but remember these are only to get you thinking and we welcome novel approaches!

Photo Classification

Maybe you've heard of our ability to identify hot dogs (and other foods) in photos. Or how we can tell you if your photo will be beautiful or not. Can you do better?




Natural Language Processing & Sentiment Analysis

What's in a review? Is it positive or negative? Our reviews contain a lot of metadata that can be mined and used to infer meaning, business attributes, and sentiment.

Graph Mining

We recently launched our Local Graph but can you take the graph further? How do user's relationships define their usage patterns? Where are the trend setters eating before it becomes popular?

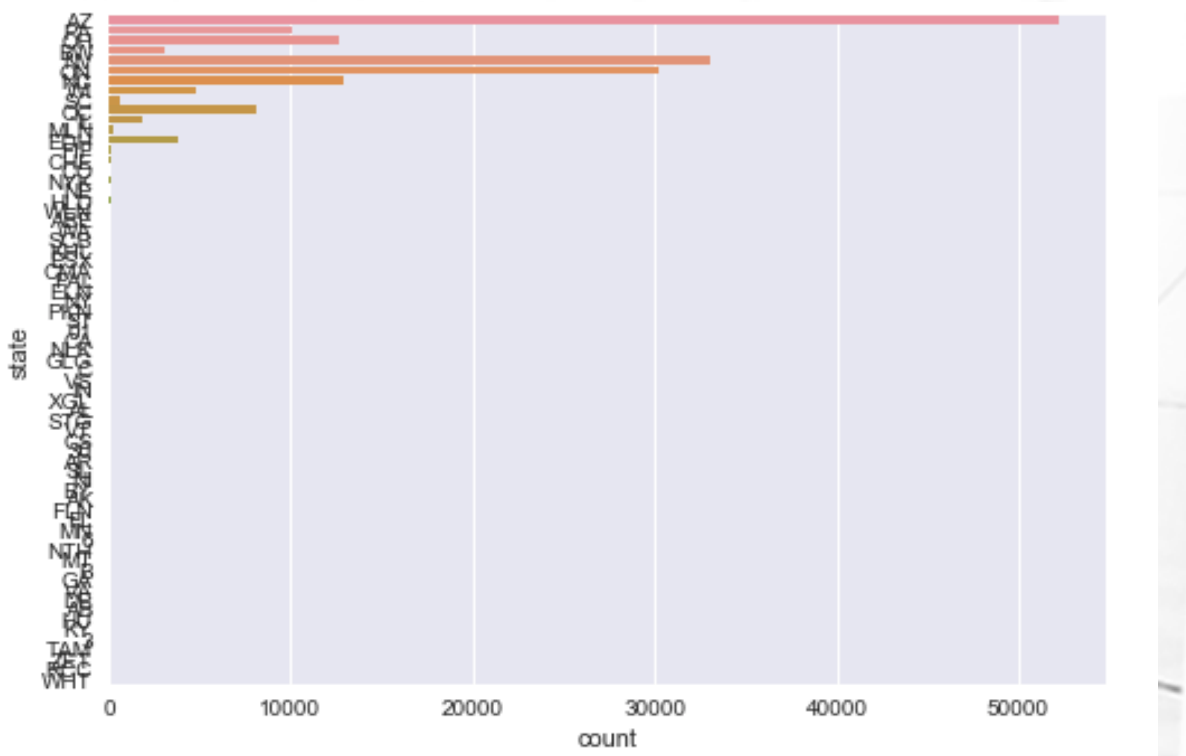
Introduction

5.26 million reviews:

business_id, review_id, user_id, date, text, stars, cool, funny, useful

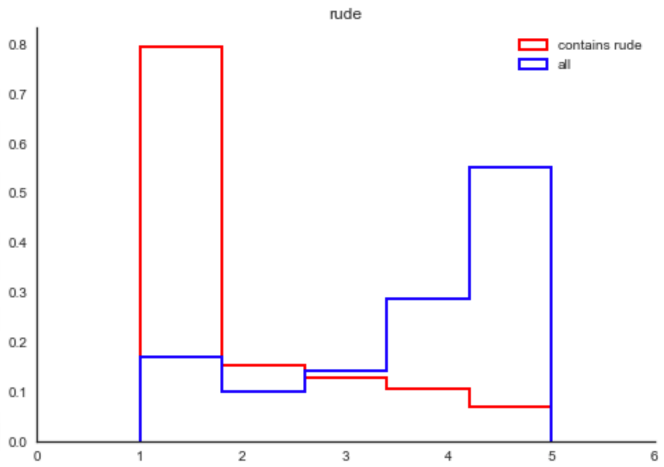
Sample subset:

- 0.866 of all business are in states: ['OH', 'NC', 'ON', 'AZ', 'PA', 'NV']
- 0.1% of reviews in these states: 4836 reviews
- text, stars

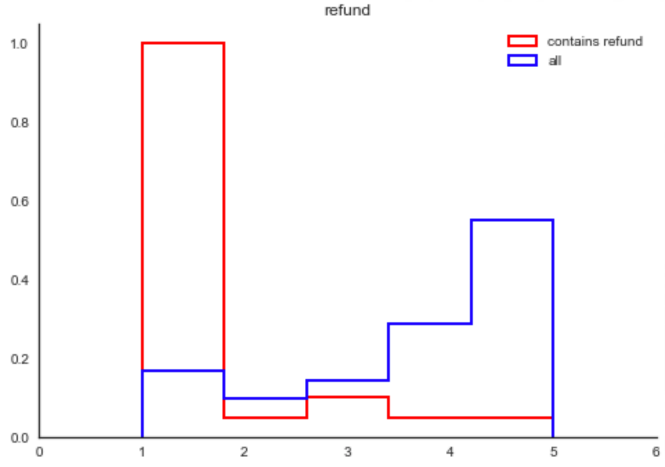


Introduction

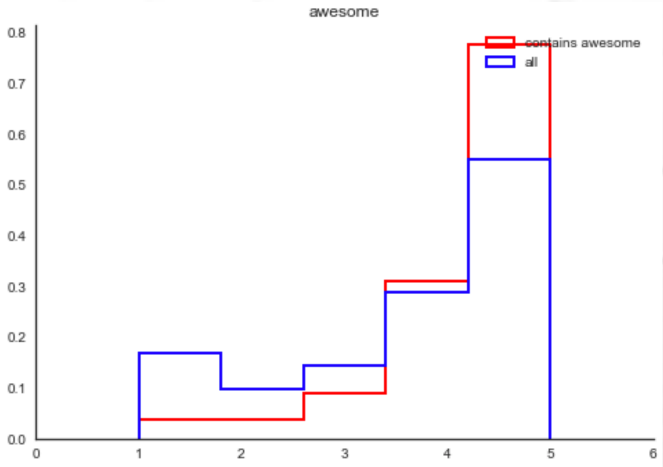
rude



refund



awesome



Introduction

Goal: classify a review as positive or negative

positive: stars 4 - 5

negative: stars 1 - 3

Excellent service and I love the chefs, very polite and take the time to talk with you. I've been at this location 3 times and have loved the food and it's been very good. The AYCE is great and they have an excellent happy hour for spirits and beer and sake.
This last visit my friend had some type of beef served on a small dish and I had a bite, but it tasted like rubber. Other than that great sushi and rolls!

Introduction

positive

negative

Data Preprocessing

- Change to lower case
- Remove punctuations
- Remove numbers
- Remove stop words
- Stem words

```
['classify', 'classifies', 'classifying', 'classified', 'classification']
['classifi', 'classifi', 'classifi', 'classifi', 'classif']
```

Feature Extraction: Bag of N-grams

- **Tokenize** strings and give an integer id for each possible token
- **Count** the occurrences of tokens in each document
- **Normalize** with diminishing importance tokens that occur in the majority of documents.

Choice of N:

e.g. “not good”

n = 1, “not” “good”

n = 2, “not good”

mixed 1&2, “not” “good” “not good”

Feature Extraction: TF–IDF

Some very present words carry little meaningful information.

TF-IDF = $TF(t,d) * IDF(t)$

TF: Term Frequency

IDF: Inverse Document Frequency

$$IDF(t) = log\frac{1+n_d}{1+df(d,t)} + 1$$

n_d: total number of documents

$df(d,t)$: number of documents that contain term

High IDF means rare in corpus

Results:
Bag of N-grams vs TF - IDF

AUC	Logistic	Linear SVM	RF
Unigram	0.835	0.820	0.760
Bigram	0.798	0.773	0.704
Mix of Unigram and Bigram	0.859	0.842	0.747

AUC	Logistic	Linear SVM	RF
Unigram	0.864	0.848	0.729
Bigram	0.801	0.790	0.688
Mix of Unigram and Bigram	0.867	0.859	0.745

All vectorizers have max_df = .95 and min_df = 3.

Linear SVM is trained with stochastic gradient descent.

AUC above is the average AUC of test data under 10 different training (0.8)– test(0.2) splits.

```
LogisticRegression with TF-IDF (mixed 1&2 grams):
Smallest Coefs:
['got minut' 'dog' 'experi realli' 'ask need' 'act' 'diesem' 'eat beef'
 'comida' 'classi' 'got meat' 'given' 'bead' 'bit high' 'appeal'
 'bland flavor' 'delici boyfriend' 'chicken order' 'best view'
 'definit thi' 'busi just']

Biggest Coefs:
['alway like' 'delici littl' 'earlier' 'car took' 'einfach' 'dong'
 'artifici' 'church' 'clean use' 'bet' 'coffe' 'extrem pleas'
 'burger flavor' 'dessert order' 'blond' 'did offer' 'alik'
 'blueberri pancak' 'ethic' 'car servic']
```

Other Try: Word Embedding

Words are represented by dense vectors where a vector represents the projection of the word into a continuous vector space.

Keras Embedding Layer

- **input_dim**: size of the vocabulary in the text data.
- **output_dim**: size of the vector space in which words will be embedded
- **input_length**: length of input sequences

Other Try: LSTM

```
model.add(Embedding(100, 128, input_length=5222))
model.add(LSTM(128))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))
print(model.summary())
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_6 (Embedding)      (None, 5222, 128)         12800
_________________________________________________________________
lstm_6 (LSTM)                (None, 128)               131584
_________________________________________________________________
dropout_2 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 129
_________________________________________________________________
activation_2 (Activation)    (None, 1)                 0
=================================================================
Total params: 144,513
Trainable params: 144,513
Non-trainable params: 0
```

Other Try: LSTM

```
CPU times: user 1h 34min 48s, sys: 46min 29s, total: 2h 21min 17s
Wall time: 47min 16s
3084/3084 [==============================] - 120s 39ms/step
Train score: 0.185312799578
Train Accuracy:  0.755188067368
343/343 [==============================] - 14s 40ms/step
Test score: 0.187066628964
Test Accuracy:  0.752186589095
```

Future Work

- Prediction - based Embedding:
    1. Continuous Bag of words
    2. Skip – Gram model

- Count - based Embedding:
    - GloVe

Thank you !