Text classifier for 400,000 Amazon reviews

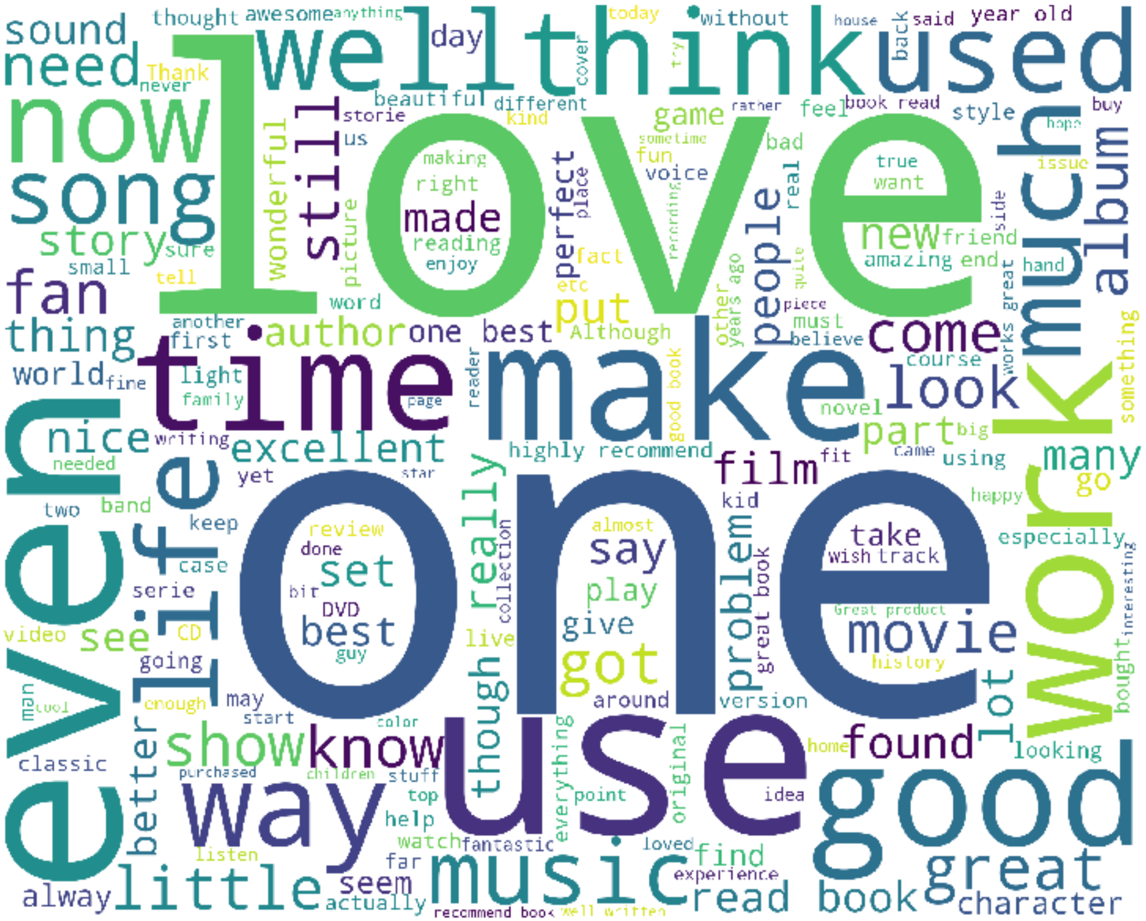
Chubing Zeng

University of Southern California

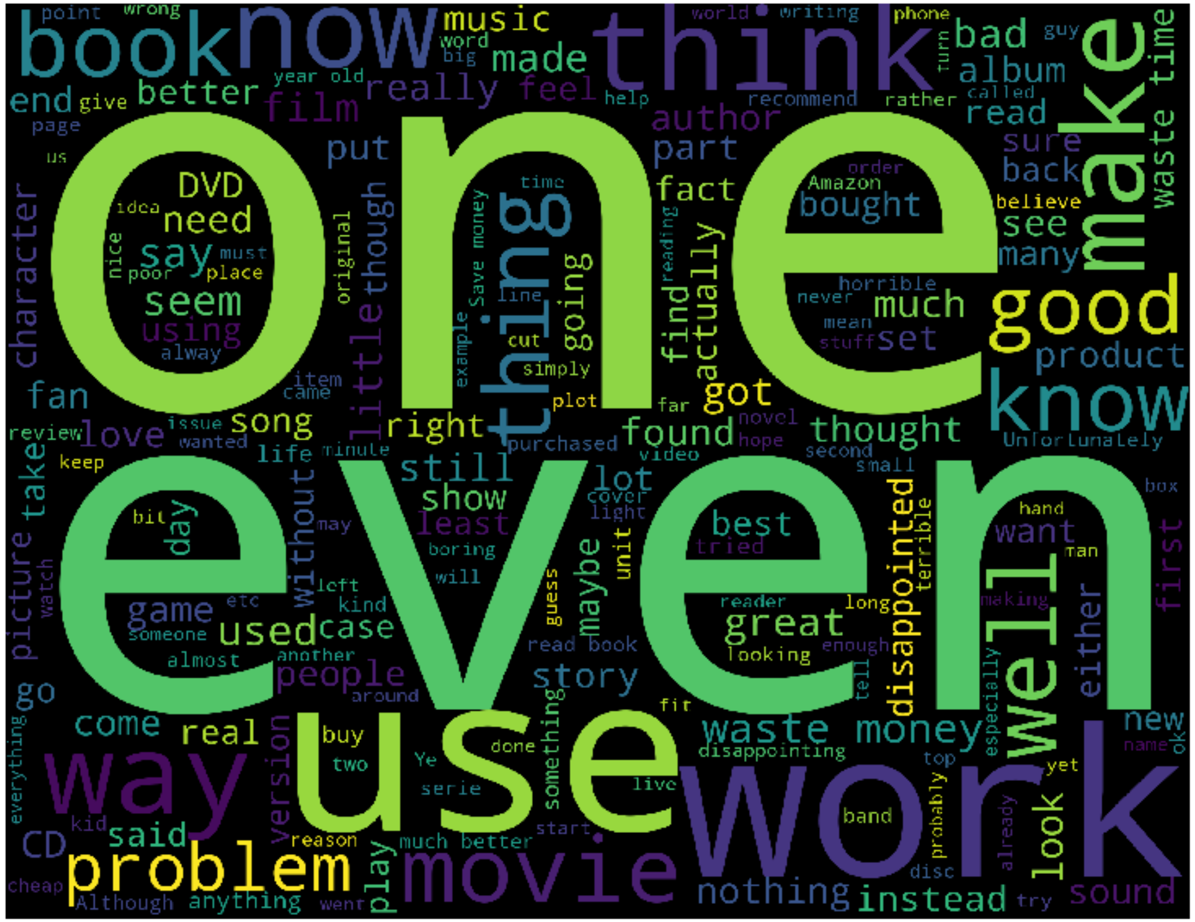
July 25, 2018

Introduction

Goal: build a text classifier to determine whether a review is positive or negative using a a dataset with 400,000 Amazon reviews

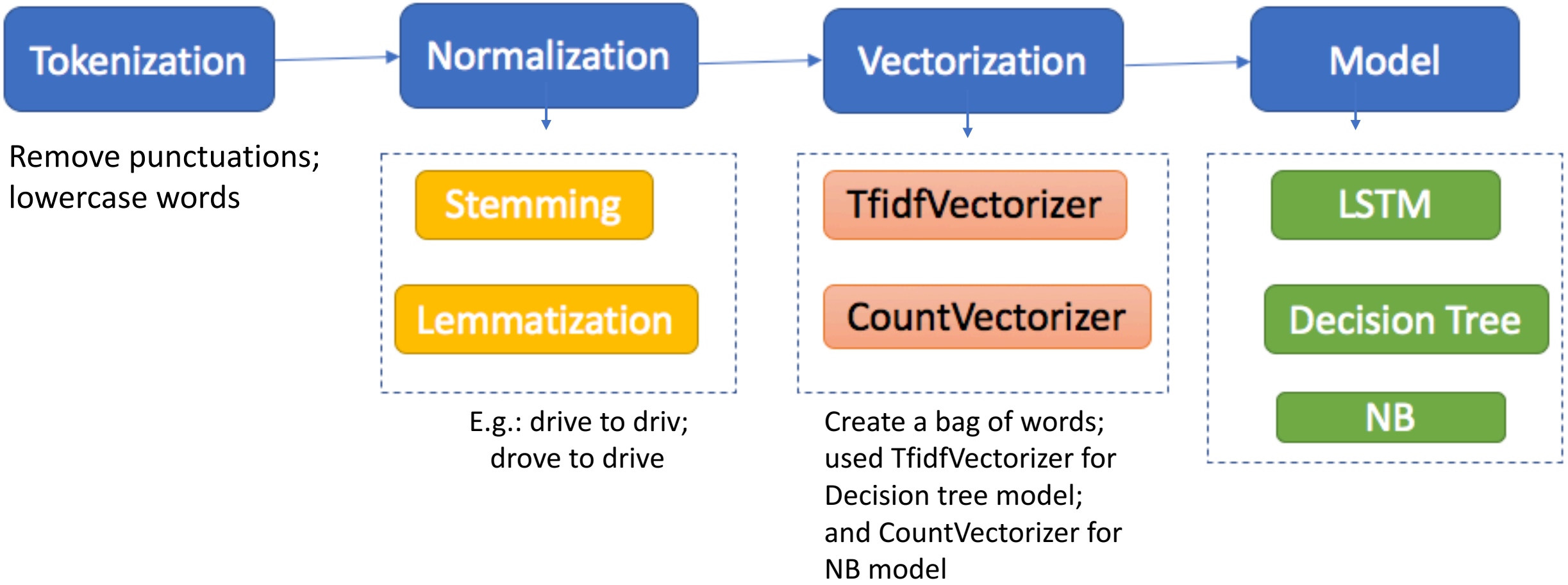


Positive reviews word cloud



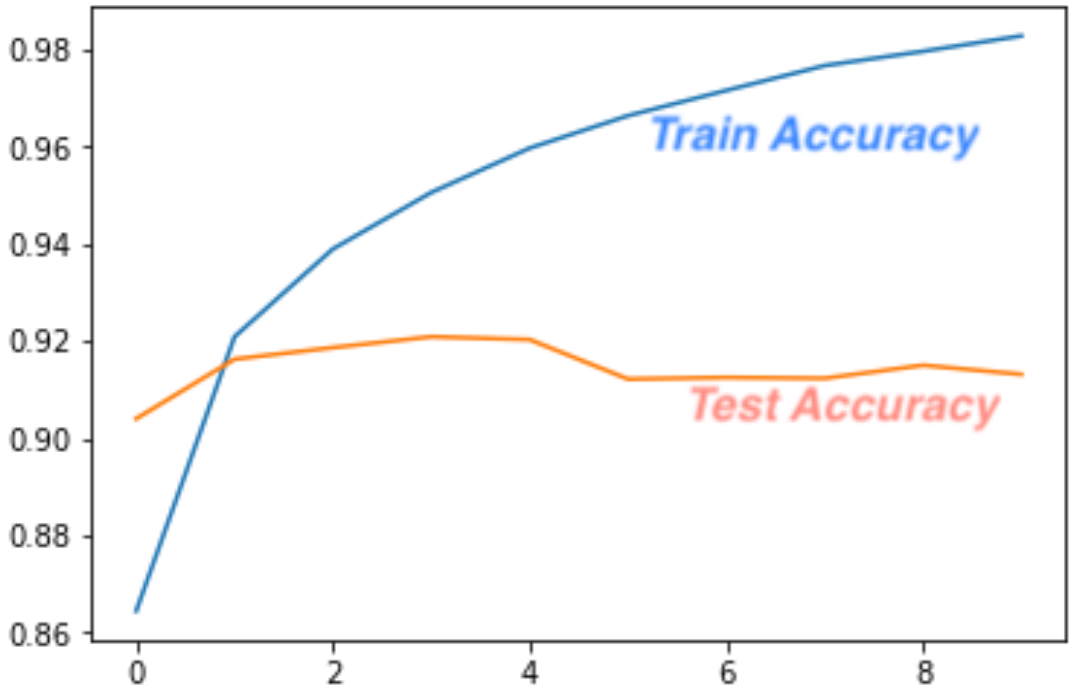
Negative reviews word cloud

Data preprocessing pipeline:



Classification methods I: Long Short Term Memory networks (LSTM)

- Used Keras to train LSTM model
- Total parameters to train: 31,319,681
- **Training time: about 3 hours** 😪 😪 😪 😪 😪




```
# ==============================================================================
# LSTM Modeling
# ==============================================================================

# Pad sequences
x_train = sequence.pad_sequences(X_train, maxlen=200)
x_test = sequence.pad_sequences(X_test, maxlen=200)
x_train.shape
#x_train = x_train[:25000]
#y_train = y_train[:25000]
model = Sequential()
model.add(Embedding(len(vocab_to_int) + 1, 128))
model.add(LSTM(128, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))

model.summary()

model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

batch_size = 600
history = model.fit(x_train,
          y_train,
          batch_size=batch_size,
          epochs=10,
          validation_data=(x_test, y_test),
          shuffle=True)
```

Classification methods II: Decision Tree

- Implemented using sklearn.tree.decisontreeclassifer
- **Training time: about 35 min** 😪 😪 😪
- Used TfidfVectorizer

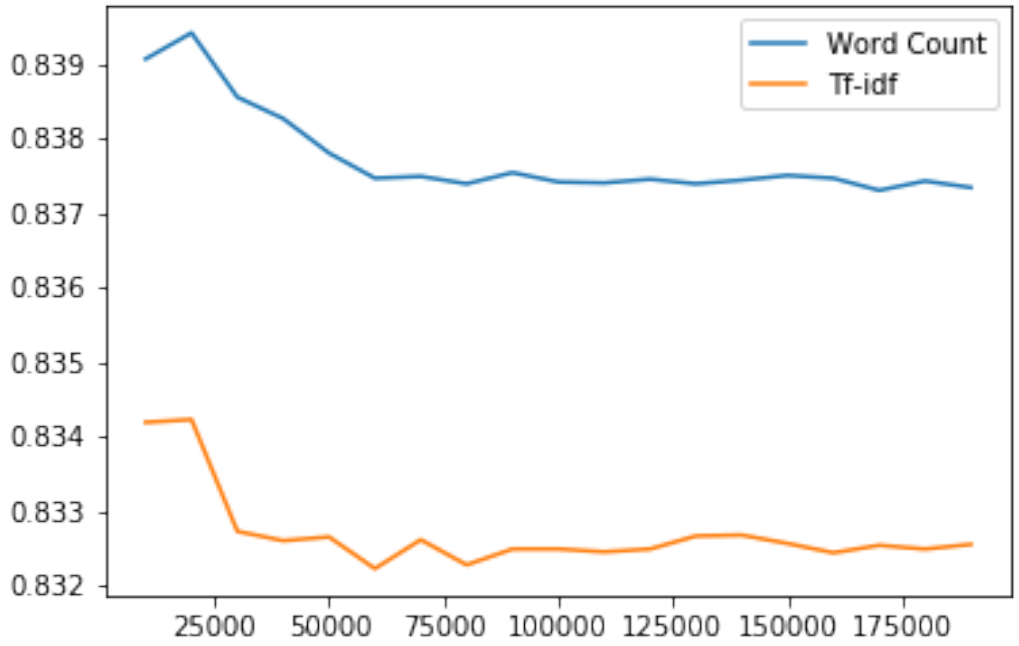
```
tf_vec = TfidfVectorizer(vocabulary=topwords)
train_features = tf_vec.fit_transform(train_x)
# ========================================================================
# Decison Tree Modeling
# ========================================================================
from sklearn.tree import DecisionTreeClassifier
dtree_model = DecisionTreeClassifier()

# Train Model
import time
start = time.time()
dtree_model.fit(train_features, train_y)
end = time.time()
print("Decision tree model trained in %f seconds" % (end-start))
```

Classification methods III: Naïve Bayes model

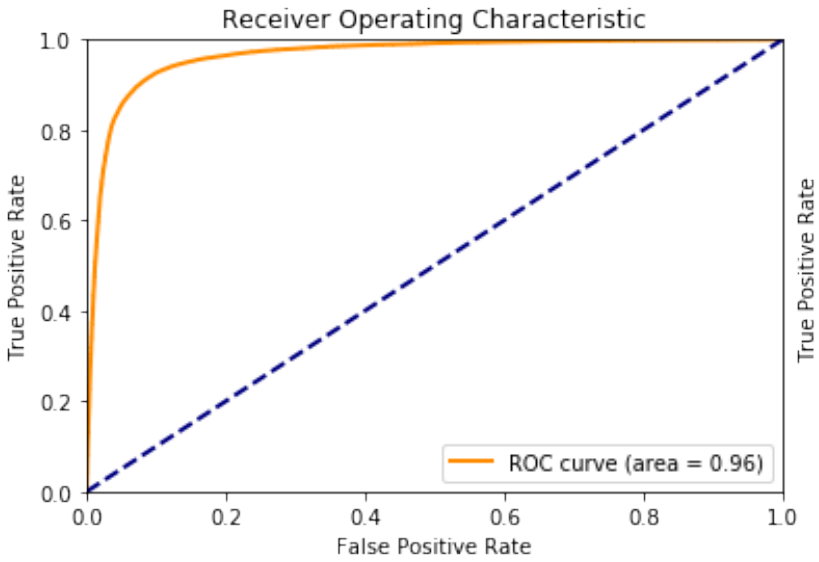
Why I chose this method:

- ✓ Very fast to train -- **training time: about 1 second** 😀 😀 😀
- ✓ Benchmark to compare with other methods (More of a statistical method than “machine learning” method)
- ✓ Relative good performance
- ✓ Easy interpretation

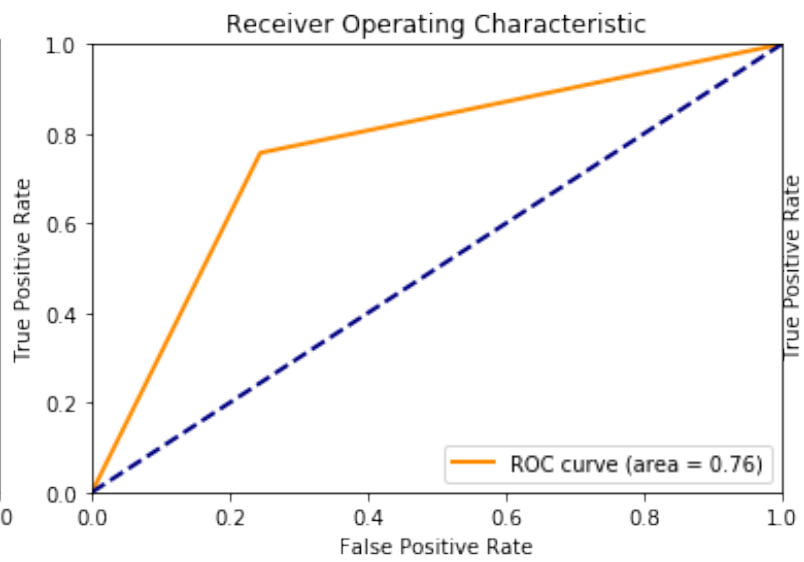


Result: ROC Curves

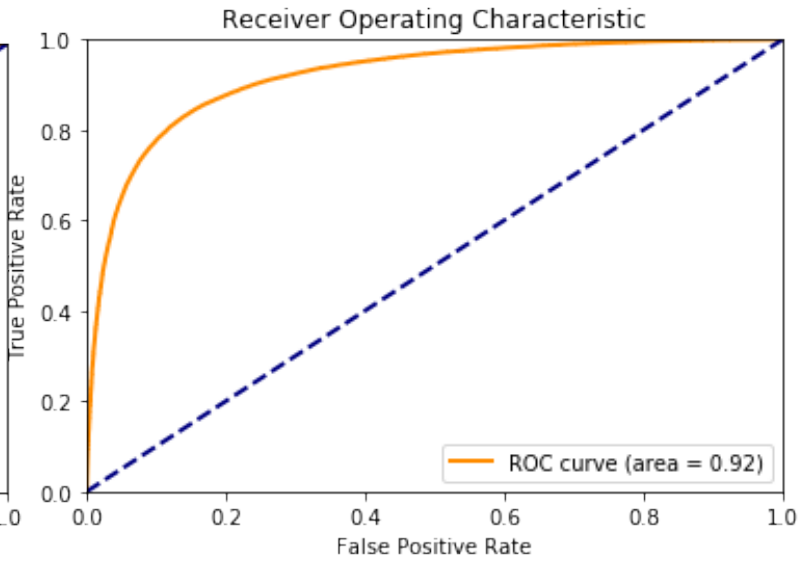
First split the review data into a training data and a test data. Every 5th sample belongs to test data, the remaining samples belong to training data. Trained using training set and validated model in testing set. Number of negative sample: 40068; number of positive sample: 39920



Neural Network (LSTM)



Decision tree



Naïve Bayes

Result: F1 Score Table

Method	Precision	Recall	F1 score	AUC
LSTM				0.96
0	0.92	0.90	0.91	
1	0.90	0.93	0.91	
Average	0.91	0.91	**0.91**😄	
Decision Tree				0.76
0	0.76	0.76	0.76	
1	0.76	0.76	0.76	
Average	0.76	0.76	**0.76**☹️	
Naïve Bayes				0.92
0	0.85	0.83	0.84	
1	0.84	0.85	0.85	
Average	0.84	0.84	**0.84**🙂	