Accuracy Analysis of Different ML Models on Amazon Fine Food Review Dataset

Masashi Omori
December 9, 2017

Problem Statement

- Online food reviews are often used for customers to assess the quality of a product.
- Different users can have different ratings based on the same experience. This makes the number of star review irrelevant.
- How can we standardize ratings based on reviews? What would be a good machine learning model to use?

Approach

➔ **Train**

Train a neural network and a couple of standard classification models to predict positivity of review based on the raw text review.

➔ **Analyze**

Check the accuracies achieved by each of the models and find the optimal hyperparameters/model.

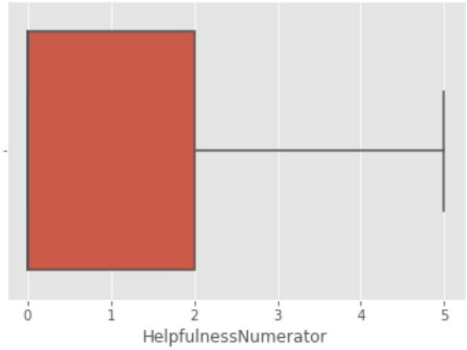
Data Preparation/Analysis

- **Data was taken from kaggle (570k reviews):**
  https://www.kaggle.com/snap/amazon-fine-food-reviews/data
- **It consists of features such as:**
  - **ProductID**
  - **UserID**
  - **Score**
  - **Text**
  - **HelpfulnessNumerator/Denominator**
    - **# users who found review helpful & # users who indiicated whether review was helpful or not**

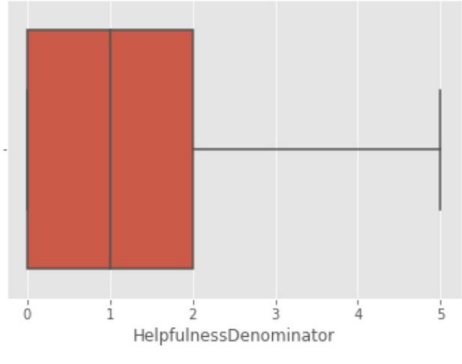
	Score	Text
0	5	I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better. My Labrador is finicky and she appreciates this product better than most.
1	1	Product arrived labeled as Jumbo Salted Peanuts...the peanuts were actually small sized unsalted. Not sure if this was an error or if the vendor intended to represent the product as "Jumbo".
2	4	This is a confection that has been around a few centuries. It is a light, pillowy citrus gelatin with nuts - in this case Filberts. And it is cut into tiny squares and then liberally coated with powdered sugar. And it is a tiny mouthful of heaven. Not too chewy, and very flavorful. I highly recommend this yummy treat. If you are familiar with the story of C.S. Lewis' "The Lion, The Witch, and The Wardrobe" - this is the treat that seduces Edmund into selling out his Brother and Sisters to the Witch.
3	2	If you are looking for the secret ingredient in Robitussin I believe I have found it. I got this in addition to the Root Beer Extract I ordered (which was good) and made some cherry soda. The flavor is very medicinal.
4	5	Great taffy at a great price. There was a wide assortment of yummy taffy. Delivery was very quick. If your a taffy lover, this is a deal.

- We check the distribution of the helpfulness numerator and denominator data and check for data quality.
- Noticed that there are clear outliers in the data, and some ratio values (numerator/denominator) is > 1.0, which shouldn't happen. These outliers and invalid data were cleaned

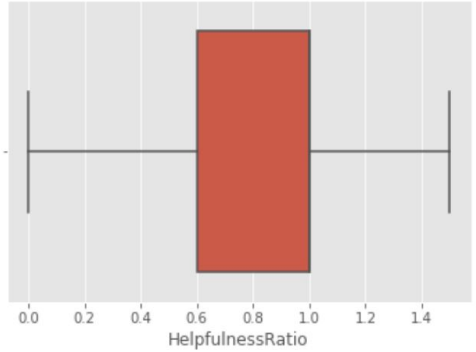
```
Helpfulness Numerator quantiles
0.700    1.0
0.800    2.0
0.900    4.0
0.950    7.0
0.990    19.0
0.995    30.0
Name: HelpfulnessNumerator, dtype: float64
```




```
Helpfulness Denominator quantiles
0.700    2.0
0.800    3.0
0.900    5.0
0.950    9.0
0.990    23.0
0.995    35.0
Name: HelpfulnessDenominator, dtype: float64
```




```
Helpfulness Ratio quantiles
0.700    1.0
0.800    1.0
0.900    1.0
0.950    1.0
0.990    1.0
0.995    1.0
Name: HelpfulnessRatio, dtype: float64
```



Machine Learning: Baseline Models

Approach

➔ **Embed**

Embed the text reviews into numerical vectors for machine learning.

➔ **Train/Fit**

Use a logistic regression and SVM to fit the embedded data and analyze the accuracies achieved.

➔ **Assess**

Use inferential statistics to create a confidence interval of accuracies to assess significance of accuracy increase in neural network .

Sequencing Representation of Texts

- Encode words into numerical vectors.
- For example, consider the sentence 'I eat an apple' with a vocabulary list of 'I', 'eat', 'an', 'apple', 'orange'. The cardinality of the vocabulary list is five, so there will be a 1x5 vector representation of each datapoint. In the above example's case, the representation will be [1,1,1,1,0]. Similarly, for the sentence 'I eat eat an orange,' the representation would be [1,2,1,0,1] (since there are 2 ‘eat’ in the sentence).

Logistic Regression

- Put the encoded vectors into a logistic regression for training/testing.
- A grid search was used to find the optimal regularization and number of iterations till convergence.
- Found that l2 norm is the best regularization, and any iterations above 10 is unnecessary.

	params	accuracy
0	{'penalty': 'l1', 'max_iter': 10}	0.813810
1	{'penalty': 'l2', 'max_iter': 10}	0.825714
2	{'penalty': 'l1', 'max_iter': 100}	0.814286
3	{'penalty': 'l2', 'max_iter': 100}	0.825714

Support Vector Machines

- Put the encoded vectors into a SVM for training/testing.
- A grid search was used to find the optimal regularization and loss function.
- Found that l2 norm is the best regularization, and squared hinge as the best loss function.

	params	accuracy
0	{'penalty': 'l1', 'loss': 'hinge'}	0.790714
1	{'penalty': 'l2', 'loss': 'hinge'}	0.805000
2	{'penalty': 'l1', 'loss': 'log'}	0.799286
3	{'penalty': 'l2', 'loss': 'log'}	0.794286
4	{'penalty': 'l1', 'loss': 'squared_hinge'}	0.783571
5	{'penalty': 'l2', 'loss': 'squared_hinge'}	0.802857

Accuracy Assessment

- Ran repeated tests with the baseline models to create a distribution of accuracies. This will be used to provide evidence of how statistically significant the accuracy improvement is in neural network training.
- The below table are the 99% confidence interval from fitting/predicting the data 1000 times. The resulting models have very little variance and shows consistent accuracy values.

model	testing accuracy
Logistic Regression	[0.83733333, 0.84466667]
SVM	[0.83666667, 0.84466667]

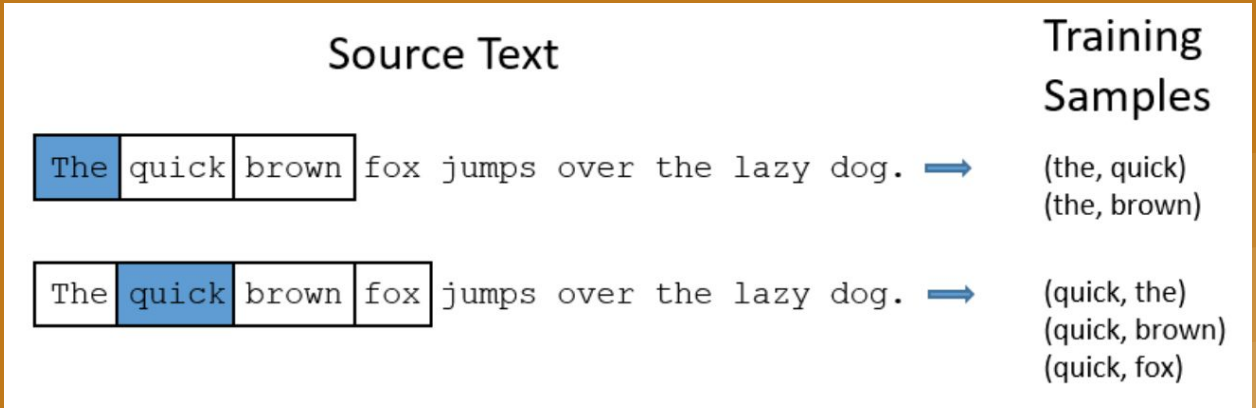
Machine Learning: Convolutional Neural Networks

Approach

- **Word2Vec**
  Embed the reviews by using a different algorithm.
- **Hyperparameter tuning**
  Find the optimal parameters for the CNN
- **Train/Test**
  Using the optimal parameters, analyze the performance of CNN

Word2Vec Encoding

- Encode words into numerical vectors.
- Instead of bag-of-words where each word is encoded with an index, word2vec stores semantic meaning of words into a vector. It does this by looking at words around a target word, and creating a probability distribution for each combination of words. For example, it can learn relationships like "Man is to woman as brother is to sister."



CNN: Hyperparameter tuning

- Once the word embeddings are in place, we can feed the data into the neural network for training. In order to improve the model accuracy, few parameters were taken into consideration to optimize.
  - Output dimension of embeddings
  - Number and size of convolution windows (kernel sizes)
  - Number of feature maps
  - Activation functions
- Neural network complexity increases drastically with certain parameters and those were taken into consideration during tuning. We want a good estimation of the target variables, but over-tuning the parameters can lead to overfitting and huge increase in training time. In contrast, under-tuning would underfit. To avoid this the hyperparameter tuning of the CNN was based off of the article "A Sensitivity Analysis of (and Practitioners' guide to) Convolutional Neural Networks for Sentence Classification." https://arxiv.org/pdf/1510.03820.pdf
- To speed up the process, 2 epochs were used for each CNN.

CNN: Hyperparameter tuning

- Visual results of the hyperparameter tuning. In short, we found the below most optimal values:
  - Output embedding dimension: 8
  - Kernel sizes: [9,9,9]
  - Number of feature maps: 700
  - Activation function: sigmoid

output dimension	training accuracy	testing accuracy
2	85.44	84.54
4	86.42	85.51
8	87.75	86.47
16	87.77	86.41
24	87.90	86.40
30	87.43	85.92

activation function	training accuracy	testing accuracy
relu	86.36	85.20
tanh	86.79	85.53
sigmoid	86.78	85.63

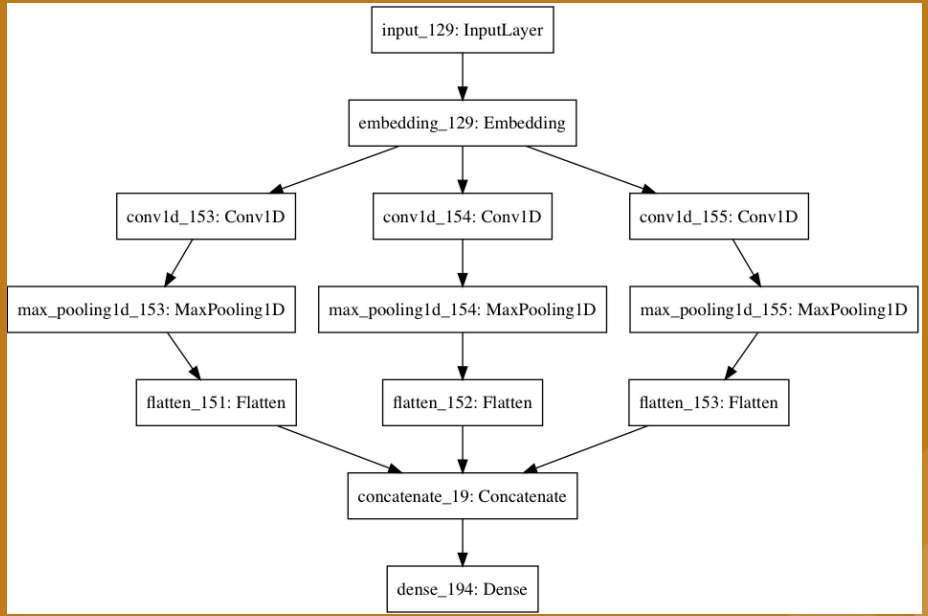
filters used	training accuracy	testing accuracy
9	88.50	86.9
9,9	88.75	87.19
9,9,9	92.01	90.56
8,9	91.31	89.80
9,10	89.56	87.69
8,9,10	91.94	90.42

number of features	training accuracy	testing accuracy
10	92.59	91.20
50	93.79	92.22
100	94.10	92.48
200	94.35	92.64
400	94.51	92.69
600	94.47	92.69
700	94.85	92.91

window size	training accuracy	testing accuracy
1	80.98	80.83
2	83.38	83.02
3	85.08	84.39
4	86.20	85.30
5	86.98	85.67
6	86.86	85.54
7	87.75	86.28
8	87.71	86.29
9	88.39	86.97

CNN: Optimal Model

- A CNN was trained using the optimal parameters found.
- For the testing data, it correctly labeled the data ***93.28%*** of the time compared to ***83%*** of the baseline models. This is a very significant increase (with 99% confidence) and shows that a correctly tuned CNN can outperform a more standard machine learning model by a very large amount.



model	testing accuracy
Logistic Regression	[0.83733333, 0.84466667]
SVM	[0.83666667, 0.84466667]

Further Notes:

- In terms of training time, a single logistic regression or SVM took less than 5 minutes on my laptop (macbook pro 8GB RAM), whereas training a CNN took close to 1 hour per epoch (2 epochs were used for a total of ~2 hours. Multiply this by the number of models created during hyperparameter tuning, it would've taken days for the process to finish).
- To speed up the process, an AWS p2.xlarge instance was used, which has a builtin GPU. Since most calculations of a neural network is matrix computation, a GPU sped this process up significantly (~10~20x). With this, training the final model took roughly 15 minutes. The total amount of time took to do the hyperparameter tuning was around 7 hours, and costed about 6 dollars in AWS EC2 fees.