Amazon Fine Food Reviews Sentiment and Helpfulness Classification

Alain van Rijn
ay.vanrijn@student.maastrichtuniversity.nl

Abstract

This work shows the application of sentiment classification and helpfulness prediction on the Amazon fine food review data-set. Logistic regression and LSTM classifiers are fitted on both tasks, using different data modelling strategies to obtain desirable performance metrics.

1 Introduction

Throughout history, word of mouth has been one of the essential sources of data for decision-making. In the modern era, online product reviews specifically, play a significant part in the purchasing decisions that are made by consumers. The original intent of the presented research was to predict product ratings from product descriptions, and review sentiment from review text the Amazon fine food reviews data-set (Project, 2017). The goal was adapted to predicting review sentiment and review helpfulness while leaving out the product descriptions. The primary reason is that the author suspects there is a giant lurking variable between the product description and its rating, namely the product quality itself. At the same time, review helpfulness prediction seems a task that is relevant to commercial application and is similar to review sentiment in its workings. One can easily imagine that a helpful review should show to a user before a less helpful one, and to accomplish this in the absence of votes can be valuable to the retailer and consumer alike. The original Kaggle competition on the data-set produced multiple classifiers for review sentiment; however, the usefulness rating in the data-set remains little researched. The author will, therefore, explore review sentiment classification, as well as establish a method of classifying helpfulness on the same data-set.

2 Approach

2.1 Data exploration

2.1.1 Data properties

Kaggle (2017) describes the dataset as follows: ”This dataset consists of reviews of fine foods from amazon. The data span a period of more than 10 years, including all 500,000+ reviews up to October 2012. Reviews include product and user information, ratings, and a plain text review. It also includes reviews from all other Amazon categories.” The following properties are known:

- Reviews from Oct 1999 - Oct 2012
- 568,454 reviews
- 256,059 users
- 74,258 products
- 260 users with $>$ 50 reviews

The data available on each review is:

- ProductId: the product the review belongs to
- UserId: the id as given by amazon
- ProfileName: the displayName chosen by the reviewer
- HelpfulnessNumerator: the number of positive helpfulness votes
- HelpfulnessDenominator: the total number of helpfulness votes
- Score: the rating from the user of the product with ProductId on discrete likert scale or star rating 1-5
- Time: timestamp

- Summary: summary of the review as specified by the user
- Text: the main text of the user review

One example of a review:

- ProductId: B001GVISJM
- UserId: A18ECVX2RJ7HUE
- ProfileName: "willie ""roadie"""
- HelpfulnessNumerator: 2
- HelpfulnessDenominator: 2
- Score: 4
- Time: 1288915200
- Summary: fresh and greasy!
- Text: good flavor! these came securely packed... they were fresh and delicious! i love these Twizzlers!

It is good to know that there are no missing data points in any of the columns. Also, there is no single user close to having more than 0.5% of the reviews under one username. In other words, there should be no issue with the structure of the data-set. The mean product rating was 4.18 stars, and the median is five stars, the mean helpfulness numerator/denominator 1.74/2.23 votes respectively. The median review in terms of helpfulness has no votes.

2.1.2 Data preparation

Each task has a similar but slightly different data preparation.

Sentiment data preparation In order to obtain sentiment data and labels, the neutral 3-star reviews are ignored. Reviews of 1 and 2 stars get positivity label 0. The four and 5-star ratings get a label of 1. This approach was commonly seen in the Kaggle submissions. For all sentiment classifiers, 75000 training and 25000 testing examples are used, due to limits in available memory.

Helpfulness data preparation In the realm of helpfulness prediction, there is fewer common ground on data preparation found in previous research. The author prepared the data based on the same principles used in the sentiment classification; the main point of it was to remove examples in the middle. Another factor in usefulness is the number of votes. After trial and error with a logistic regression classifier it was decided to only keep training examples with at least 7 votes and helpfulness ratios $\frac{hnum}{hdenom} < 0.33$ or $\frac{hnum}{hdenom} > meanhelpfulness$ where $meanhelpfulness \approx 0.7828$. The former category gets Helpful label 0, and the latter a 1. In this way, only examples persist for which there is a strong opinion on the helpfulness of each review. Out of all reviews, 31847 examples remained after applying the filter of which 5310 had a label of 0. This pruning has the advantage of reducing the number of training examples and therefore, the memory requirements while keeping the most discriminative data. Care must be taken that the false positive rate remains acceptable since approximately 83.3% accuracy could be achieved by a one rule classifier that classifies all examples positive.

2.1.3 Text preprocessing

Two configurations of text-processing were used throughout the experiments, one configuration without processing and another that does lower-casing, Html tag removal, punctuation removal and snowball stemming (sno). The reason to use the text processor would be the mapping of different words with the same meaning into one, so the number of unique tokens is decreased, which should lead to more data per unique token. In other words, text processing should reduce sparsity. The choice for the specific processing set-up is inspired by observing different Kaggle submissions. During an experiment with the Logistic regression classifier, it was perceived that the text processing made close to no difference in the performance of the model, hence not all text-processing - classifier combinations are tested. The text processor used is indicated for each result.

2.2 Modelling strategies

Both helpfulness and sentiment classification uses the (filtered) review text column of the data-set as their input data X, but have their own set of output labels y as described in 2.1.2. The text needs to be represented in some way such that the classifier can handle it. For each classifier, a different way of representation is appropriate.

2.2.1 Logistic regression

The sklearn logistic regression implementation was used on the helpfulness and sentiment classification task alike. The input review text is transformed into a bag of words representation by the sklearn Countvectorizer. The counts of meaningful units of text are recorded in a sparse vocabulary vector. A decision to make here is the n-gram lengths to consider. Making only the use of unigrams would imply that only individual tokens are represented without regard to their history. A sentence containing ”not good” as a substring would then have ”not” and ”good” both counted once, regardless of their position. Considering bigrams would solve this case. Ngram sizes 1 to 4 to captures phrases such as ”not nearly as good”, which gives more possibilities for a model to assign longer distance word relationships. The Logistic regression model itself learns logistic odds of an example belonging to one class and a cut-of point that makes the outcome binary in our case.

2.2.2 LSTM

The LSTM gets input represented as word sequences. The Keras fit_on_text function first creates a vocabulary based on the words occurring in the input training set. The texts_to_sequences then converts the text of each example into sequences of word indices as defined in the vocabulary created by the fit_on_text function. The superpowers of the LSTM come from the fact that it can relate information from early in a sequence to outputs that occur much later in the text. This happens by nature of the LSTM. Default vocabulary size of 5000 and max sequence length of 500 is used throughout. The dropout rate, by which nodes are deactivated randomly during training to reduce overfitting, is set to 0.2 during each experiment.

3 Experiments

3.1 Sentiment classification

Logistic regression sentiment classification The Logistic regression classifier on sentiment classification was run with different n-gram lengths to show the relevance of n-gram length on this task. For helpfulness prediction by logistic regression. The best settings. The maximum number of features was unrestricted on this task so equal to the n-gram vocabulary sizes. All test set-ups for this task are without preprocessing. The highest area under the curve was obtained with the uni till trigram representation, where improvements from the uni till bigram where already negligible. 4-gram was the largest n-gram tried. The maximum number of iterations was set to 1000, which was enough to converge in all cases.

sentiment logistic regression				
max_ft	ngarm	preproc	AUC	FPR
-	1 till 1	none	0,8423	0,2808
-	1 till 2	none	0,8856	0,2065
-	1 till 3	none	0,8856	0,2073
-	1 till 4	none	0,8853	0,2083

Table 1: experimental results sentiment classification by logistic regression

sentiment LSTM			
LSTM layers	accuracy	epochs	preproc
1	93,73%	5	snowBall
2	91,98%	3	snowBall

Table 2: experimental results sentiment classification by LSTM

LSTM sentiment classification With the text modelled as described in 2.2.2, both a single and a 2-layer LSTM where used in this task for 5 and 3 epochs respectively. The single layer LSTM had the highest accuracy score of the two.

3.2 Helpfulness

Logistic regression helpfulness classification Using the most successful parameters from the sentiment classification task, the helpfulness prediction was made in a similar manner with data modelled as described in 2.2.1. Tf-idf word embedding was briefly tried but found inferior. The notable difference is that the uni till unigram and uni till bigram was left out. Also, restrictions in memory imposed some limitations on the maximum number of features for this task. The results show no striking discrepancy between 2^{15} and 2^{16} maximum allowed features. In trail experiments, increasing the maximum number of features did improve results up until about 30000 features. Using the snowball prepossessing hardly influenced the model performance. The uni till 4-gram model performed the best overall by a small margin. The false-positive rate did not change significantly in the different settings shown.

LSTM helpfulness classification The same default settings were used as in the sentiment

helpfulness logistic regression				
max_feat	ngarm	preproc	AUC	FPR
2^15	1 till 4	none	0,8101	0,3356
2^15	1 till 4	snowBall	0,8104	0,3356
2^16	1 till 4	snowBall	0,8107	0,3379
2^16	1 till 3	snowBall	0,8097	0,3401

Table 3: experimental results helpfulness classification by logistic regression

helpfulness LSTM				
LSTM layers	accuracy	epochs	preproc	FPR
1	89,53%	5	snowBall	0,3412733
2	89,55%	3	snowBall	nan
2	88,72%	3+5	snowBall	nan

Table 4: experimental results helpfulness classification by LSTM

classification task 3.1 with the language model described 2.2.2 adding an extra LSTM and training for three epochs layer proved only marginally helpful, despite increasing the training time noticeably. Five more epochs even decreased the test accuracy from 89.55% to 88.72% while the accuracy during training after eight epochs was about 98% vs the 94% training accuracy observed after three training epochs with the same model. The false-positive rate was similar although slightly higher than that with the Linear regression model, despite increased overall accuracy.

4 Results

4.1 Sentiment classification

The experimental results 1 and 2 show how each setting performed on the sentiment classification task. The largest absolute jump in the power of the logistic regression model was observed when changing the maximum n-gram size from uni-gram to bi-gram. This can plausibly be explained by the fact that sentimental meaning is often found in two-word combinations as well as single words. The phenomenon that increasing n-gram size after that helped so little was somewhat surprising although it can be due to larger n-grams being simply too sparse in just 75000 training examples. The LSTM, with its penchant to do well on sequential data, produced the most accurate results. Adding an extra layer somewhat reduced the test accuracy, and should probably only be done when using a larger amount of computational resources, as typically learning more complicated functions requires more data to happen satisfactorily.

4.2 Helpfulness classification

Helpfulness classification or prediction results are shown in 3 and 4. Besides accuracy, attention should likely be paid to the unhelpful reviews classified correctly. It would be strongly beneficial to consumers not having to read unhelpful reviews. The preparation of the data-set resulted in having an unbalanced data-set in which roughly 83.3% of the reviews are helpful. All the most promising classifiers shown in the experiment results managed a false positive rate of roughly a third or conversely two-thirds of the unhelpful reviews correctly classified. An Extra LSTM layer gave about the same accuracy, and performing additional training epochs lowered the test accuracy, likely due to overfitting on too little data. A larger, more balanced training-set could resolve this problem. In the case of the fine-food reviews, there are no additional reviews available that meet the filtering requirements stated in 2.1.2. Another possibility would be only to keep the topmost helpful reviews equivalent in size to the bottom least helpful. Assuming the least helpful group would stay the same, this would leave us with only about 10000 examples. On the subject of data selection, one could also argue, that the moderately helpful reviews ought to be represented since otherwise, a real-world classifier might not have the proper training to classify those.

5 Conclusions

In sentiment classification, we have seen results in the same order of those published in the relevant Kaggle competition. It was found that preprocessing did not meaningfully improve the results, but using longer than uni-grams and enough features in the representation did help logistic regression to perform better. It would be interesting to see if future work could predict neutral reviews on top of the positive and negative ones.
With a few tweaks to the data, helpfulness could be predicted by the same algorithms that classified sentiment. The insightful metric of false positives showed that the classifiers did better than random chance or a majority classifier alone. The LSTM did yield better accuracy but was not able to solve the false positives any more than logistic regression. The false-positive rate is suspected to be a

data problem. More data and modelling data differently could improve future results. Humans tend to disagree on which reviews are helpful. The disagreement is reflected in the data and therefore, in the prediction results.

References

Snowball.

Stanford Network Analysis Project. 2017. Amazon fine food reviews.