Building Models For Identifying Helpful Reviews Using The Amazon.com Reviews Dataset

Arjun Bakshi (bakshi.11)

Abstract

Online reviews of products enhance the ability of users to discern the quality of the product being sold. However, the quality of the reviews themselves varies greatly, and as a result, so does their utility. This project uses the amazon.com reviews dataset to identify simple textual features that are correlated with the helpfulness of reviews, and aims to build a model that can classify reviews as helpful or not based on features learnt.

1. Introduction and motivation

As more products and services become available for purchase online, online reviews of these products become more important. Filtering the useful reviews from the fluff would reduce the burden of reading multiple reviews before making their decision. A few websites like amazon.com allow users to vote on reviews, recording whether they thought the review was helpful or not. However, a lot of websites do not have such tools in place that could be used to filter reviews. It would therefore be useful to study the characteristics of reviews posted online to learn to distinguish between reviews that are useful and reviews that aren't. The amazon.com dataset is a perfect candidate for such a supervised learning exercise as it already contains a large number of reviews that people have vote on[1]. Another advantage of such insight would be identifying reviews that have been posted by the seller to artificially boost the rating of the product they are selling, as such reviews should most likely be marked as unhelpful by customers. This project takes a simple approach, described below, to the problem of identifying helpful reviews based on patterns of keywords, keyphrases, and parts of speech used in the review text.

2. Approach

2.1 Keyword / N-gram model

The idea here is that readers look for certain keywords or key phrases when reading a review of a product. The mere presence of a keyword of interest may answer a question in the reader's mind regarding the product, thus making the review helpful. The same logic can apply to key phrases, and combinations of keywords, or combination of keyphrases. As these phases can be of variable length, n-grams for will be extracted for a range of values of n. This model implies that the absence of these keywords makes a review less helpful.

2.2 Parts-of-speech N-grams model

The idea here is that although the exact words may change, the sentence structure underlying helpful reviews may be the same. The aim of the POS N-gram model is to see if this underlying structure exists and can be extracted from helpful reviews. For this purpose, the dataset will be

transformed into POS tags, and run through the same sort of analysis as the Keyword/N-gram model.

3. Goals

- Examine the truthfulness of the claim that users look for certain keywords, and that the presence of these keywords make reviews more helpful.
- Study the effects of “n” for keyphrase based models.
- Compare the effectiveness of POS models vs the keyword models

4. Dataset

The dataset being used for this project is the amazon.com reviews dataset which contains ~35 million reviews collected over 18 years[1]. The dataset is also available by category of products. Preliminary analysis of some categories reveal the following information relevant to this project:

- A large portion of reviews have no votes, or very few votes on them. This reduces the number of usable reviews from millions to a few thousand in all categories under examination.
- Of the remaining reviews, a very large percentage of them are very helpful reviews (>90% of the people found it helpful). This means that a baseline model that predicts all reviews to be helpful will do much better than a random guess.

Instead of using the entire dataset, reviews for only the following subsets will be used:

- Shoes
- Musical Instruments
- Arts
- Cell phones & Accessories
- Watches

These categories were chosen because of their narrower scope. Categories like Software, Sports & Outdoors, and Electronics are too broad and will make the task of keyword selection more difficult. For example, models build on the entire dataset may be biased towards keywords from larger product categories as they may occur more frequently. Separating the data and training different models for smaller and more specific categories will result in better tuned models.

There is also a bias towards smaller datasets because of limited processing power. Other datasets may be examined as time and computing resources permit.

5. Implementation

5.1 Libraries used

The following python libraries were used to implement the ideas and conduct the experiments:

Numpy: Matrix and scientific computing library for dealing with matrices.[2]
Scikit Learn: Machine learning library for building classification models using Random Forest, and for building the document-term matrix using vectorization.[3]
NLTK: Natural Language Toolkit used for tokenization, Part-of-Speech tagging, and Stemming of review text.[4]

5.2 Data preprocessing

The general flow of the preprocessing is as follows:

1. An amazon.com dataset is converted to column form with one column each for text, helpful votes, and total votes.
2. Conversion of that file to POS tags or stemmed words if needed.
3. N-gram extraction and vectorization to convert a review sentences to a <review-n-grams-vector, review-class> format, and feature pruning.
4. Classification of reviews using Random Forest Learners.

The original dataset files were converted to a column format, with one column for the title and text of a review, one column for the total number of votes on the review, and one column for the total number of helpful or positive votes on the review. This is a one time transformation and various input files can be built on top of these files.

These files were then used to generate separate files for POS tagged reviews, and files where all the words in the reviews were stemmed. Stemming was done to unify the frequency or count of words with the same root word across the corpus.

Each of these files was then run through vectorization code. The vectorization code converted a review into a vector, where each position in the vector represented a word in the dataset vocabulary, and the value at that position represented the number of occurrences of that word in that review.

Obviously, the total vocabulary of the dataset is very large and so, the following conditions were applied as pruning measures:

- Words that occur in less than 50 documents were discarded.
- Words that were not unique enough to extremely helpful, or extremely unhelpful reviews were discarded.

5.3 Feature selection

For each word, the total number of its occurrences in the dataset, and its distribution of occurrences across reviews of different levels of helpfulness was computed. If the fraction of times it occurred in reviews with helpfulness score of >=90% was greater than or equal to 0.8, or less than or equal to 0.2, it was kept as a word of interest. Otherwise, it was discarded. This step reduced the length of the vector at the end of vectorization, without affecting the accuracy of the models built on it. This was verified by generating pruned and unpruned vectorized datasets and examining the classification accuracy of models built on them.

6. Experiments and Results:

6.1 Experimental setup

For each dataset, three column form files were created. One with the original text, one with POS tagged text, and one with stemmed text. NLTK was used for tokenization, POS tagging, and stemming. The Porter Stemmer from the NLTK library was used for stemming.

After the creation of those files, 5 different models were built on each file. The models differ in the size of n-grams extracted from the file. The different values of n chosen were 1,2,3,4, and [1-4]. Under the [1-4] setting, n grams of all four sizes are extracted together and used in the same model. This is in contrast to the first four settings where a separate model was built for each n.

An extracted n-gram is kept or discarded based on the pruning logic explained in section 5.3. If kept, the n-gram becomes a feature in the dataset sent to the Random Forest classifier.

Class labels are binary, helpful or not helpful. Each review in the dataset has a number of helpful votes, and a total number of votes on it. Using these two numbers, we can compute what percentage of people found a review helpful. For these experiments, the reviews are divided into two classes based on this percentage value. Reviews that are helpful to less than 90% of the people are labeled as “unhelpful”, and the others are labeled “helpful”. This threshold though seems arbitrary, produces the best performance, and divides most datasets in a 60:40 ratio. Other thresholds were tested, and the proposed models showed no improvement or degradation compared to the baseline for those thresholds.

The Random Forest classifier was chosen because of its speed of operation, and high classification accuracy when compared to RBF-SVM, KNN, and Naive Bayes classifiers. 30 estimators are used in the Random Forest classifier. The Orange Canvas data mining tool was used for the preliminary analysis for finding the optimal type and settings of a classifier for this problem[5].

The classification accuracy of the models was averaged over 10 iterations. The data was split 70:30 between the training and testing set. The baseline model accuracy was computed by predicting the majority class for each test review.

6.2 Comparing the Keyword and POS models

The results are presented in terms of absolute percentage gain or loss over the baseline in classification accuracy averaged across all 5 datasets. The standard deviation for all experiments over 10 iterations is below 1%. The results for different models are shown in table 1.

As seen in table 1, the classification accuracy of the models built on the original text and the stemmed text decreases as the length of the n-grams increases. This can be explained by

considering that as the length of the n-grams increase, n-grams become more diverse, and not many are frequent enough to be accepted into the list of valid features for the classification task. As a result the classifier has to learn to distinguish classes using fewer and lower support features, resulting in poorer performance. It can also be inferred from the identical performance and trend of these two models that they are similar in the type of information they use to learn the classification model. That they are are fundamentally similar to each other, and fundamentally different from the POS model.

N-gram size	1	2	3	4	1 to 4
Orig. text	9.25%	8.75%	1.85%	0.20%	11.39%
Stemmed	8.38%	9.15%	1.69%	0.41%	11.30%
POS	-0.03%	1.06%	2.54%	4.68%	6.16%

Table 1: Average absolute % improvement in classification accuracy. Baseline accuracy, 59.66%, is the same for all settings.

For the POS models, the classification accuracy increases with longer n-grams. This can be explained by examining the pruning function used during vectorization. Shorter n-grams are more frequent, however they tend to be equally frequent across all types of reviews. As a result, most n-grams are discarded by in the pruning step. This leads to the same problem as seen in the case of the original text and stemmed text models where the classifier does not have enough features to learn a good classification model. However, as the length of n-grams increases, the diversity increases. Unlike the case of the other models, there are only 34 unique POS tags, and so the problem of over diversification and low frequency or support for longer n-grams does not prevail. As a result, the number and quality of features available to the classifier increases as the length of the n-grams increases. Intuitively, this effect is valid only over a range of small values of n, as the effects of over-diversification will become apparent for much larger n.

For all three models, it is seem that combining n-grams of different length results in the best models. This implies that models built on different length of n-grams capture different information, and contribute unique and helpful features to the composite model.

In general though, the models based on the original text and stemmed words outperform the POS models. But it is worth mentioning that the POS features could improve the performance of the other models if POS features are added to them.

6.3 The pruning metric, and keyword to review helpfulness relation

The feature pruning, or keyword selection mechanism described in section 5.3 is examined for its validity. To do this, from each original dataset, a new dataset is made with no pruning or

feature selection performed. Random Forest classifiers are learnt on these datasets and the performance is compared to the pruned version in terms of improvement over baseline in classification accuracy. The number of features in each dataset is also recorded.

	No pruning		Pruning	
Dataset name	+ %	# features	+ %	# features
Arts	5.90	1227	6.39	405
Music Inst.	12.02	4364	13.70	311
Watches	12.86	2374	14.44	224
Shoes	6.51	876	7.92	131
Cell phones	14.29	7152	14.51	89

Table 2: Classification accuracy improvement with and without pruning

As seen in table 2, the models built using the features selected by the pruning metric perform marginally better than, or at least at the same as, the model built using the complete feature set. It also does so while using a small fraction of the complete list of features. Since the pruning metric selects only those n-grams that are unique to very bad or very good reviews, this result can be seen as evidence that users look for keywords or key phrases in reviews, and their vote on the review is positively correlated with the presence of these keywords.

7. Conclusion

This project presents a simple and intuitive idea for filtering helpful reviews from unhelpful ones. The results of the experiments performed make it clear that there is some merit in the intuition proposed in this work. This work also examines the effects of the length of n-grams on predictive models, and different types of text transformations, like POS tagging and stemming on model performance. Further examination could be done on the effect of combining POS tagged, stemmed, and original text features.

8. References

[1] J. McAuley and J. Leskovec, “Hidden factors and hidden topics: understanding rating dimensions with review text.” RecSys, 2013.
[2] Walt, Stéfan van der and Colbert, S. Chris and Varoquaux, Gaël, “The NumPy Array: A Structure for Efficient Numerical Computation.” Computing in Science & Engineering, 2011.
[3] Pedregosa et al., “Scikit-learn: Machine Learning in Python”, JMLR 12, 2011.

[4] Bird, Steven, Edward Loper and Ewan Klein, "*Natural Language Processing with Python*." O'Reilly Media Inc., 2009
[5]Demšar, J., Curk, T., & Erjavec, A. Orange: Data Mining Toolbox in Python; Journal of Machine Learning Research 14(Aug):2349−2353, 2013.