Data Analysis and Artificial Generation of Amazon.com Product Reviews

John Feilmeier

4.1: Source files

Source files, both in CSV and ARFF format, are available at my project's Github repository[1]. The raw data from which I am generating these CSV and ARFF files (`review_data_big_raw.csv`) has been turned in via `make turnitin` on acad.

4.2 Initial Documentation

4.2a: Data Source

The source of my dataset is customer reviews of various products on Amazon.com. A list of URLs of the products' review pages is available in the `dataRetreival` section of my project repository[2]. I selected these products by searching Amazon.com using free association, and sorting the results by those items with the most ratings using the `Amazon™ Sort - Number of Reviews` plugin[3] for Google's Chrome web browser.

4.2b: Goals

As one of the most-used e-commerce companies in the U.S.[4], Amazon.com provides millions of customers[5] across the nation with numerous products and services, and many of these customers leave reviews of the products they have purchased. These reviews can be very important[6] to a product, as shoppers with access to many similar items will choose those with better ratings. This provides the interested data analyst with a huge repository of labelled natural language data. My goal in analyzing this data is to find consistencies in the similarly-rated reviews for different products, and consistencies in the similar products for high or low-rating reviews. These similarities may then be helpful in automatic text generation using an LSTM neural network. Another more personal goal is to expand my knowledge of Natural Language Processing and generating content with Long Short-Term Memory neural nets. This is an interesting field that has many applications in user interface design and machine learning, and this project is an opportunity for me to dive in and get some experience with it.

The data from Amazon.com product reviews is a popular one, and has been explored for various[7] outcomes[8]. I scraped my data from Amazon, so my product reviews are unlikely to match the datasets used by others. As far as my knowledge of other projects, I am beginning a new analysis of this data.

4.2c: Steps Taken So Far

To begin with, I retrieved the amazon review data (product name, rating, review title, review text, etc.) as a JSON file, using techniques from two web-scraping tutorials[9]. At first, I was only getting a few reviews for each product, leading to a small dataset and poor initial results. I found the second tutorial/ guide (referenced at [9]) and added some of that functionality to the initial script. The change in URL caused some of the previous code to break, so I had to inspect the HTML in ChromeTools to find the proper location of the Title and Price information. I then wrote a python script to parse the JSON data and write it to a raw CSV file[10].
The next step was to try to figure out what useful information I could coax/squeeze/beat out of the 'bag-of-words' contained in the raw data. Using NLTK[11] and some incredibly helpful tutorials[13] about NLP and machine learning in Python, I wrote some functions to generate two classes of output. The first is simply a CSV file with the most common words as headers, each review row giving a 'True' value if it contained the word. The second involved some more extensive processing with the NLTK library. The review text is tokenized and lemmatized[14], and the results are then checked for spelling errors and labelled to find the correct part of speech. The list of tagged words is then processed to retrieve the percentages of nouns, verbs, etc. that make up the review or the review title. Using the english stop word list from NLTK, percentages of stop words or meaningful words are also generated.

~~Challenges~~ Opportunities

The less complex process has two parameters that can change the nature of the output. Looking into the resulting files, I saw some oddly specific words that had made it into the top 200 words, and so added the ability to turn each review into a Python 'set' before processing. This prevents reviews with heavily repeated words from distorting the most-used words across all reviews. This created a somewhat changed dataset. I also added the option to use the tokenized review, which simply parses a review into a list of unchanged words, or the lemmatized review, which uses the NLTK lemmatizer to convert the words into some calculated root word.
While the NLTK library is indispensable, I am also unsure of how effective some of the stemming and lemmatizer functions are at producing good data. Working with a large dataset, it is a challenge to go through and actually read all of the results to make sure they make sense. That is the main reasoning behind having 4 options for the simpler program. If I can generate a few different permutation of the data, hopefully one produces reliable models and I can find out why it did and the others didn't.
The spell checking is based on the Enchant port for Python[15]. I have not mastered its application yet, as a cursory glance at the results show a fairly high percentage of misspelled words. My own preconceptions about users' literacy and the effort to correct accidental spelling mistakes leads me

to think that a good measure of spelling accuracy will be helpful in classifying reviews, but I am also interested to be proven wrong.

4.2d: Commercial Applications

The results of this analysis could be used in any area where some qualitative text must be generated for an item or experience. Ethical and legal considerations aside, companies could generate spam positive reviews for their products that could pass for human-created text, pushing their product rating up and to the top of the results for searched keywords on e-commerce websites. The results could also be used in Research and Development to see what customers are focusing on when they leave positive or negative reviews, providing feedback to the producers of the product about where there is success and where there is room for improvement.
This same process, if not overfitted to product reviews, could be used for other human-written forms as well. As automation begins to replace much of the mundane work of form completion, many forms still require a written textual account. If a dataset of previous written textual accounts could be analyzed, much of this extra work could be eliminated, freeing up the employee's time for more valuable pursuits.

4.2e: ML and modelling Techniques

My goal is to predict whether a review is positive or negative, or for one product or another, so I anticipate most of the analytical techniques used on this dataset to be nominal classification. For the 'bag-of-words' approach, my initial thought is that a Naive Bayes-type of algorithm will give a good result, as none of the attributes (the common words) are necessarily related to each other. For the morte complex dataset, I think some sort of Decision Tree or Decision Table will be my best bet, as the structure of the data I generated from the raw state shows statistics about each review, those values having been generated from the string and their values intertwined. I also plan to throw this data into a LSTM neural net and see what happens.
WEKA, Scikit-learn data-mining package for Python, and Keras with both TensorFlow and Theano backends will provide me with the data analytics platforms I will use for this analysis. RapidMiner may be used for data visualization, as it has more advanced capabilities than WEKA and is simpler than the python library Matplotlib to use.

4.2f: Other Interesting Items

While I do have a dataset prepared, I am curious as to whether one set of reviews may be more predictive than others, or if there is some product with reviews that are more representative of the general population's reviewing. My choice of products was entirely arbitrary, and this choice was made to remove the likelihood of product-specific words from being counted as very important. I may find that a single product with many reviews, such as the Kindle Fire tablet, can provide a better model to apply to the other random reviews.
Another aspect of this project that grabs me is that the raw data, while useful as a dataset in its own right, can be processed in many ways to possibly provide a more useful dataset. While I may not

unlock the unanswered secrets of Natural Language Processing, I will be able to objectively judge the efficacy of my parsing of the review data, as the simpler 'bag-of-words' and my more heavily processed data face off in the various machine learning algorithms at my disposal.

References:

[1]: https://github.com/Jacktavitt/product_review_fun/tree/master/source/dataParsing/parseResultFiles
[2]: https://github.com/Jacktavitt/product_review_fun/blob/master/source/dataRetreival/product_review_all.dat
[3]: https://chrome.google.com/webstore/detail/amazon-sort-number-of-rev/hepimngelnnmpbpklphhbbmalefoploi
[4] : https://www.cnbc.com/2018/01/03/amazon-grabbed-4-percent-of-all-us-retail-sales-in-2017-new-study.html
[5]: https://www.statista.com/statistics/546894/number-of-amazon-prime-paying-members/
[6] : https://www.entrepreneur.com/article/253361
[7]: http://worldcomp-proceedings.com/proc/p2015/DMI8036.pdf, L. Jack and Y.D. Tsai
[8]: http://math.arizona.edu/~hzhang/math574m/Read/mineMillionsReviews.pdf, Kunpeng Zhang, Yu Cheng, Wei-keng Liao, Alok Choudhary
[9]: https://www.knoyd.com/blog/amazon-review-scraper and https://www.scrapehero.com/how-to-scrape-amazon-product-reviews/
[10]: https://github.com/Jacktavitt/product_review_fun/blob/master/source/dataRetreival/scrape_reviews_big.py, from https://www.scrapehero.com/how-to-scrape-amazon-product-reviews/ with changes by John Feilmeier
[11]: https://www.nltk.org/
[12]: https://pythonprogramming.net/tokenizing-words-sentences-nltk-tutorial/
[13]: https://pythonprogramming.net/tokenizing-words-sentences-nltk-tutorial/
[14]: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html, © 2008 Cambridge University Press
[15]: https://github.com/rfk/pyenchant