Capstone Proposal

Hannah Moran

November 15, 2017

Proposal

Domain Background

Sentiment analysis / opinion mining and topic modelling are two of the most useful applications of natural language processing and machine learning, particularly when used to analyze the large volumes of human-generated opinions and advice on the internet. Websites that aggregate user reviews help people make decisions on everything from what to eat for dinner to which plumber to hire to which mattress to buy, and can expect to generate revenues by hosting these reviews, whether by driving sales of their own products and services or simply through advertising revenue.

As the body of work around sentiment analysis, also known as opinion mining, for online reviews has grown, the approaches to performing it have become more sophisticated in their ability to handle problems that are specific to this particular type of text - it's often short, full of abbreviations, symbols, idioms, non-standard punctuation, and slang, and it is highly contextual. VADER (Valence Aware Dictionary for sEntiment Reasoning) and the paper the accompanies it[1] are a great example of efforts to understand and compensate for many of the problems that plague short bits of human-generated text that are centered around opinions.

Topic modelling is another important application of machine learning to natural language texts. Latent Dirichlet Allocation (LDA) is one of the more current and mature approaches to this task, using a generative, probabilistic approach to identifying "topics" within a corpus or a document, and determining which topics are present in new, previously unseen documents.

[1] C.J. Hutto. E.E. Gilbert. 2014. VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text. Eighth International Conference on Weblogs and Social Media (ICWSM-14). Ann Arbor, MI, June 2014. http://comp.social.gatech.edu/papers/icwsm14.vader.hutto.pdf

While online reviews are incredibly useful, a number of factors can come into play that make them potentially less valuable or more difficult to interpret. First, the sheer volume of reviews can become overwhelming, while the rating is too simplistic. Many sites that host reviews provide some sort of “highlights” section, and some have searchable reviews in case you have a particular concern, but it can still be difficult to get a sense of how you might expect your own experience to go without reading many reviews. In addition, a long-running product or service might have so many reviews that recent changes in the quality of the experience might not be obvious at first glance, even if they are consistent in recent reviews - overall ratings are usually an average over all time, and reviews are not always presented in chronological order (thanks to another effort to improve their usefulness - the community-driven “helpfulness” ratings). Finally, all these issues can affect not just the users who are in the decision funnel, but also the owners of the business, product, or service who might be attempting to gather feedback from online reviews.

Problem Statement

Humans are currently better than computers at extracting meaning from natural language. However, when a corpus (collection of documents) such as a set of online product reviews grows large, this becomes a hugely time-consuming effort. Reviews often include a summary rating value which can provide a way for a person to immediately grasp high-level meaning (good, neutral, bad), but there is an opportunity to use machine learning to collect and present more information to a human reader.

If:

- we can use natural language processing and machine learning to extract “topics” (i.e., multinomial distributions over a vocabulary) from a corpus of online reviews

by:

- repeatedly minimizing the “distance” between a latent, posterior probability distribution and a simpler parametric distribution that we are using to approximate that posterior, and
- maximizing the likelihood that new, unseen text samples were “produced by” those topics (i.e., drawn from those distributions),

then:

- we can say the algorithm is learning to extract topics from natural language text.

Further, if:

- we can treat the newly extracted topics as features

by:

- assigning weights to them and repeatedly adjusting those weights until we have an algorithm that will output the likelihood that a review with a particular topic mix would correspond to a given star rating,
- maximizing the number of correct assignments the algorithm makes versus the true user-assigned rating,

then:

- we can say the algorithm is learning to use topics to understand how a user felt about a product or service, and that these topics are useful in understanding the user's opinion, given that we assume a star rating is a summary indicator of the user's views on the experience.

Finally, we can determine if using extracted topics as features improves an algorithm's performance on the star rating prediction task over simply looking at the full text of the reviews (a "bag of words" approach). Put another way - this would show that not only is the topic-learning successful, but it represents an improvement on a simpler approach to the task of understanding what kind of opinions a review expresses.

My initial intent for this project was to work with the public Yelp dataset on something I started to notice a couple years ago when I first moved to San Francisco. Often I'd eat great food somewhere with a friend only to search for the restaurant later and discover it had surprisingly low reviews. More often than not, a lot of the bad reviews would be about service, rather than food, and frequently there would be a collection of bad reviews about waiting for hours in line to eat brunch (for example), where the only negative part of the experience appeared to have been the wait. Essentially, I realized that there were probably a lot of restaurants with great food that I was passing over because of their seemingly poor 3-4 star rating, when that rating was driven primarily by things that were irrelevant to me (such as waiting for hours to eat brunch) or that were overly negative from my perspective (I personally care much less about poor service if the food is great). In researching this project I discovered that Jack Linshi very successfully explored exactly this kind of task in his 2014 Yelp competition submission[2], so I will instead attempt to further explore this idea using a collection of Amazon.com reviews for fine food products.

Much of the valuable information within a body of online reviews may be obscured as the number of reviews becomes too large for a person to read through in any practical way. Specifically, it can be difficult to:

[2] Jack Linshi. 2014. Personalizing Yelp Star Ratings: a Semantic Topic Modeling Approach). Yale University. https://www.yelp.com/html/pdf/YelpDatasetChallengeWinner_PersonalizingRatings.pdf

1. Understand quickly what aspects of their experience with the product or service users have discussed in their reviews (e.g., food quality, shipping speed, etc.)
2. Determine what the pros and cons of a product or service are - a star rating does not reveal this, and it can take too long to comb through a large number of reviews. For example, was there terrible frustration with the packaging of an otherwise excellent product?
3. Quickly become aware of any trends in recent ratings and reviews (e.g., upward ratings trend with reviews mentioning new management, downward ratings trend with reviews mentioning a new product formulation)

Datasets and Inputs

I will be using a dataset of user reviews for 74,258 fine food products from Amazon.com produced between 1999 and 2012. There are a total of 568,454 reviews from 256,059 unique users in the dataset, which I obtained this dataset from Kaggle[3] but it was originally published on SNAP[4]. The dataset was produced by Julian McAuley and Jure Leskovec at Stanford University; they collected the publicly available data from Amazon's website using a web crawler[5].

Each item in the dataset is a review, with a corresponding unique ID, a ProductID, a user ID and username, title and review text, an overall rating for the product as well helpfulness ratings for the review, and some metadata such as datetime information.

Topic modelling and sentiment analysis are difficult to use on small datasets, so I will start by restricting the analysis to the 302 products that have at least 200 reviews. Using cleaned content from the raw reviews themselves, topic modelling can be performed to understand what the users are writing about. The overall product rating (between one and five stars) gives a starting point for training a sentiment analyzer, without the need for a labelling exercise.

The dataset is also representative of other bodies of such reviews in many ways - the reviews vary widely in terms of length, how "clean" the text is, and as always the positive class (good reviews) far outweighs the negative class (bad reviews). So, any approaches that are effective here should generalize well to other similar review collections (e.g., IMDB movie reviews, etc.).

---

[3] https://www.kaggle.com/snap/amazon-fine-food-reviews
[4] http://snap.stanford.edu/data/web-FineFoods.html
[5] Julian John McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In Proceedings of the 22nd international conference on World Wide Web (WWW '13). ACM, New York, NY, USA, 897-908. http://i.stanford.edu/~julian/pdfs/www13.pdf

Finally, given that the Amazon product and user bases have only grown since 2012, and the format of reviews on the site has not changed significantly, it would be relatively straightforward to expand the dataset if needed - for example, if not enough products with a “critical mass” of reviews are available.

Solution Statement

To make a large collection of online reviews more useful to the user, can we:

1. Use topic modelling to classify the main topics of the review (e.g., service level, food quality, shipping speed), measuring the performance of the learner by looking at perplexity - the likelihood that a held-out sample of text came from the topic model the learner has constructed
    a. Using LDA topic modelling, produce a group of topics for each product with low perplexity and low overlap in key contributing terms (i.e., the top 3-5 words between topics do not have much overlap)
    b. Extend Jack Linshi’s modified LDA “codeword” approach into the Amazon reviews to produce more sentiment-aware topics, using VADER to insert codewords that indicate sentiment intensity and expose the relationship of that sentiment with the topics the LDA learner is discovering
    c. Explore performing the topic modeling on subsets of reviews to get even more granular, specific topics for individual products or groups of like products
2. Understand the success of the topic modelling by approximating the quality of the topics extracted (i.e., if they accurately reflect the opinions and sentiments of the review authors). To do this, use the topics as features to perform prediction of star ratings.
    a. Are the topics a better predictor of star rating than the raw text alone?

Latent Dirichlet Allocation (LDA) s an unsupervised method for identifying “topics,” or co-occurring groups of words, in a corpus of texts. I will be using either the gensim or sklearn LDA model, both of which are based on a python script by Matthew Hoffman[6]. This is an online variational Bayes algorithm for LDA. What does this mean, briefly? “Online” refers to the fact that it is designed to work with chunks of documents, updating itself with each new chunk as would happen if the algorithm were deployed to be updated periodically as new documents are produced from an online source (e.g., a

[6] M Hoffman, DM Blei, F Bach. 2010. Online learning for latent dirichlet allocation. Advances in Neural Information Processing Systems 23, 856-864.
http://papers.nips.cc/paper/3902-online-learning-for-latent-dirichlet-allocation.pdf

ratings/reviews website). “Variational Bayes” refers to the way we are approximating the density function of a posterior probability, by using a simpler distribution. The “Dirichlet” in LDA refers to the use of a Dirichlet distribution to model the prior probability of topics. To set out some common terminology, aligned with one of the primary papers outlining the LDA approach[7]:

- A ***word*** is discrete piece of data
- A ***document*** **w** is a sequence of *N* words, written as **w** = (w_1, w_2, …, w_N) where w_n is the *n*th word in the document
- A ***corpus D*** is a collection of *M* documents, written as ***D*** = ($\mathbf{w_1}$, $\mathbf{w_2}$, …, $\mathbf{w_M}$)

In topic modelling, we are interested in understanding the probability that a given document was generated by a topic *T.* LDA is a generative model, meaning we are assuming that the topics, as latent unobserved variables, generate the documents that we are able to observe. Given that we treat words and topics as random variables, this can become a Bayesian inference problem, where the probability of the sample corresponding to a given topic as the posterior we are trying to discover. From looking at the corpus, we can discover other necessary pieces of the equation including the probability of topic occurrence within a corpus, probability of word occurrence within a corpus, and probability of word occurrence within a topic. Eric Jang gives an excellent overview of variational inference on his blog[8].

Given that topic modelling is unsupervised, it is more difficult to judge its performance than with a supervised learner where performance can be measured on a test set with known labels. There is a metric called perplexity, discussed further below, which indicates how likely new, unseen test data is to have come from the learned topic distribution. “Quality” of the topics in terms of how well they reflect what the user was discussing in their review is much harder to quantify and usually requires some level of manual review by a human in this type of research.

However, since we are interested in getting topics that have the same function as a star rating (they give a user quick insight into the general consensus of other users), we can use a supervised learning method to look at the relationship between the two. Specifically, if we treat the sentiment-aware topics as features for the classification task, can we do a good job of predicting the overall star ratings? Jack Linshi took this approach in his Yelp research and found that the topics from his modified LDA

---

[7] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. J. Mach. Learn. Res. 3 (March 2003), 993-1022. http://www.jmlr.org/papers/volume3/blei03a/blei03a.pdf

[8] Eric Jang. A Beginner's Guide to Variational Methods: Mean-Field Approximation. August 2016. http://blog.evjang.com/2016/08/variational-bayes.html

approach outperformed the codewords generated by traditional LDA in a regression analysis against star ratings, so I would expect to find the same if the additional modification with sentiment-aware codewords is an effective one.

Benchmark Model

Natural language processing typically involves some level of human intervention in order to judge its efficacy and accuracy, whether it is an extensive labelling exercise or careful inspection of outputs to determine their quality. Additionally, it is more difficult to benchmark NLP ML techniques because their performance and tuning is so context-specific. I will need to use several approaches to benchmark the work:

1. Compare the outputs of three variants of LDA topic modelling:
    a. “Vanilla” LDA
    b. Jack Linshi’s modified LDA with sentiment (good/bad) codewords
    c. Extension of modified LDA with sentiment intensity codewords (using VADER to determine the intensity)

    The outputs can be compared using a perplexity metric but will also require some manual, qualitative review that is more descriptive in nature. A common approach in papers exploring these techniques is report on some example instances where the technique worked well as well as some where it didn’t and provide a discussion of the reasons for variance in performance.

2. Compare ability to predict a review’s star rating based on sentiment analysis of the text alone vs. by using the sentiment-intensity-aware topics (LDA variant c from above), as a way to indicate whether the topic modelling is extracting useful information for understanding the review.

Evaluation Metrics

Perplexity

There are currently no metrics that can express how “good” the topics coming out of something like LDA are - human judgment is still required to do this (and people don’t even agree all the time on this sort of thing). There is however a statistical measure called perplexity which is used to *compare* different topic models - perplexity in the case of LDA measures the likelihood that the model generated a given text sample. A lower perplexity value indicates that the model is better at predicting the test sample. So, essentially perplexity can be used to show whether various topic models “fit” a text

sample better (in the sense that they are more likely to have generated that text), but it cannot tell us anything about whether the topics make sense or are valuable to a human reader. Given this, I will use perplexity to assess the comparative quality of the three LDA variants I outlined above (and to guide hyperparameter tuning), but will need to provide discussion based on my own manual review as to the quality and relevance of the topics being generated.

Perplexity, in mathematical terms, is the geometric mean of the inverse marginal probability of each word in the held-out set of documents (the test set). A simple way to formally define perplexity is

$$\text{perplexity } (D_{test}) = exp\{ - \frac{\sum_{d=1}^{M} \log p(w_d)}{\sum_{d=1}^{M} N_d} \}$$

where $\sum_{d=1}^{M} \log p(w_d)$ is the sum over the corpus of the log-likelihood of each document (given our beliefs about the distributions of topics and words) and $\sum_{d=1}^{M} N_d$ is the sum over the corpus of the total number of keywords in each document.

Accuracy, Precision, and Recall

Evaluating the prediction abilities are much more straightforward. A simple accuracy score can easily tell us whether simple text analysis or sentiment-aware topics are doing a better job predicting star rating. For this data, we have a class imbalance problem: there are many, many more 4- and 5-star ratings than ratings with 3 stars and below. So, simple accuracy alone will not be enough to judge the performance of the classifier - a naive classifier that always predicts 5 stars no matter the feature inputs might still get a relatively high accuracy score, despite the fact that it is classifying none of the examples that correspond to lower star ratings correctly. So, precision and recall will also be required to compare these two models.

Accuracy (for a classification algorithm), commonly expressed as a percentage, is the proportion of examples in the test data set that the algorithm correctly assigned to their respective star ratings. It helps us judge, overall, how good the algorithm is at predictions. Values closer to 100% are obviously better.

Precision is the proportion of true positives to the number of positive assignments the algorithm made: (true positives) / (true positives + false positives). In this context, we will examine it on a class-by-class basis, with the class being the star rating value. The precision score would be lower than 100 for a given class if the model has predicted

most of the true positives correctly, but has also incorrectly assigned many other examples to that class. This is to say that if we have a naive classifier, it will have a lower precision score when we are looking at the dominant classes 4-stars and 5-stars.

Recall is the proportion of true positives to the actual number of positives in the dataset: (true positives) / (true positives + false negatives). The recall score will be lower than 100 for a given class if the algorithm is “missing” on many of the examples for that class - in our naive classifier example we would expect to see a low recall score for the 1-, 2-, and 3-star classes.

Project Design

1. Cleanse the dataset by performing various preprocessing tasks on the review text data, including casing, removal of punctuation and stop-words, stemming, extraction of nouns. This will produce several different variants of the text on which to perform analysis (for example, VADER can use casing and punctuation, and there may be multiple sets of stop-words that we might wish to use). Cleansing and processing will leverage NLTK packages.
2. Conduct some exploratory data analysis - number of reviews, number of products, distribution of product review counts, average length of reviews, etc.
3. Perform LDA topic modelling on the product with the largest number of reviews as well as one with only the threshold number of reviews (starting at 200) to tune the approach and determine the best version of cleaned text to use (e.g., stemmed, nouns only, etc.) as well as what the appropriate threshold number of reviews is. May need to implement a threshold for number of negative reviews given the class imbalance. All LDA implementations will make use of either gensim or sklearn depending on performance (both use the same basic algorithm).
4. Construct an implementation of Jack Linshi’s modified LDA codeword approach - each time a positive or negative word appears in a review, insert the codeword “GOODREVIEW” or “BADREVIEW.” VADER will be used to determine positive and negative words (without use of the sentiment level, for the moment). Run LDA topic modelling on the same test products to tune the approach.
5. Construct an extension of the LDA codeword approach - insert intensity codewords (e.g., “VGOODREVIEW” vs. “GOODREVIEW”) into the review text. Run LDA topic modelling on the same test products to tune the approach.
6. Compare perplexity and observed quality of the outputs of the three LDA approaches and determine if there is one that clearly outperforms the others.

7. According to results of above steps, run the LDA approaches on the full product population (above the review count threshold(s)) using the best-performing methods and report on results (perplexity and observed quality of topics output by the model(s))
8. Additional approaches to explore for extraction of review topics, depending on quality of results of the above, might include:
    a. Researching and testing ways of dealing with class imbalance in text data, e.g., an analog for upsampling with quantitative features
    b. Using clustering techniques as an alternative approach to topic modeling
    c. Using topic modeling on subsets of reviews (e.g., reviews with 3 stars or below vs. 4+ stars) as an alternative way to tease out the topics that are more prominent in negative vs. positive reviews
    d. Exploring Latent Semantic Analysis (LSA) as another approach to extracting topics and/or measuring their quality (e.g., comparatively between LDA and LSA topics)
9. Use one or more classification algorithms (e.g., logistic regression) to compare the prediction power of the raw review text vs. the extracted topics as features to predict the star rating associated with a given review; draw some conclusions about the usefulness of the extracted topics and how they justify a star rating
10. Outline additional areas for exploration based on the results