OPINION MINING USING YELP REVIEWS

Daniel Fernandez / Aldo Marini / Shounak Purkayastha

ABSTRACT

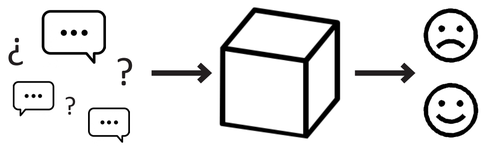
An important part of an opinion is its sentiment. To predict these sentiments, we use a Yelp dataset to train an Opinion Mining model. We explore two approaches for creating features: uni-gram bag of words model and pre-trained embeddings. The bag of words model is based on the count of lemmatized words in each opinion. The pre-trained embeddings are built using the GloVe model over 2B tweets. Both models extract semantic information from the database. Stars are assigned by users to business reviews. We first partition these stars into positive and negative sentiment. We then focus on predicting the sentiment of a review based on a balanced set in terms of users, businesses, positive and negative reviews.

INTRODUCTION

Sentiment analysis can be applied in almost every management and social domain because opinions are central to almost all human activities as key influencers of our behavior. Our behavior is, in a large part, determined by other people's attitudes and perceptions on us. The importance of opinions permeates people, businesses, governments, products and, services, to name a few [Liu 2012].

However, a significant obstacle is the sheer amount of reviews a business can receive of its services [Pakand Paroubek 2010]. This is exemplified in the popular location review app Yelp, in which a single business can have over 7,000 reviews. If no sentiment field is available, such as like/dislike or stars, then it becomes infeasible to assess whether a comment has a positive or negative sentiment without an automatized procedure. This is where machine learning comes in.

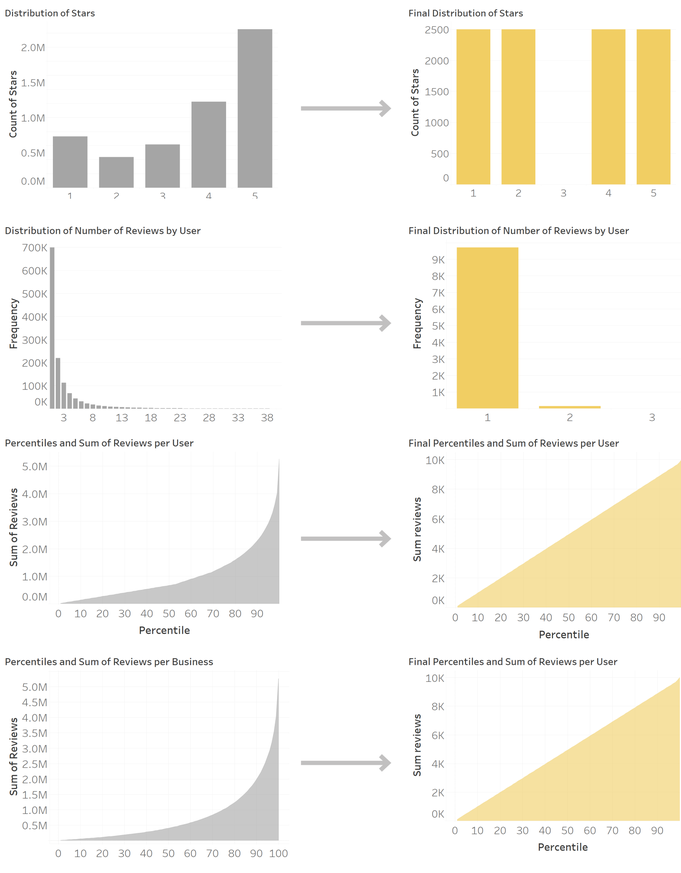
Using statistical learning techniques, we aim to take advantage of Yelp's dataset to train a general model for any kind of short reviews of products or services where a sentiment field is not available. This model will help users of untagged text to assess the opinion of a review.



DATA UNDERSTANDING

The dataset we will use for training our general review sentiment model is made available by Yelp for the Yelp Challenge. This challenge's objective is that students use their data in innovative ways and break ground in research. The dataset consist of 7 JSON files with information about businesses (business id, location, name, category, business hours, and business attributes), users (user check-in, stars given by a user to a business, review, review date), and user's friends.

Of these JSON files, we use the dataset containing the reviews, with the following variables:
user id, stars and reviews. It has 5.2 million data points. We take 1 and 2 stars as negative sentiment and 4 and 5 stars as a positive sentiment [Bo Pang 2002]. The data has imbalances in the number of stars, number of reviews per user, and number of reviews per business. To avoid domination by a small number of prolific reviewers, we imposed a limit of 40 reviews per user (which is the number of reviews of the top percentile) [Bo Pang 2002]. Additionally, to avoid domination of some business, we limited the number of reviews per business to 100. Finally, to address the imbalance of the reviews and the computational burden we take a random sample of data points of stars 1,2,4 and 5, each of size 2,500.



METHODOLOGY

We worked with reviews left by users on the Yelp site. We defined the sentiment of a review on the basis of the stars left by the user. In line with [Bo Pang 2002] we take 1 and 2 stars as negative sentiment and 4 and 5 stars as a positive sentiment and use it as the outcome variable. Reviews with 3 stars were labelled as neutral and excluded. As mentioned in previous sections the reviews were not uniformly distributed among the stars. The reviews were skewed towards positive reviews. In order to combat this and to account for computational challenges, we considered a sample of the master dataset which had an equal split of positive and negative reviews. To select the models, we evaluate discriminative, generative and tree based models and compare their cross-validation accuracy to determine the best model family. We also select the best parameters of the models by comparing their cross-validation errors.

Baseline Model

We used a baseline model that classify the reviews based on a majority vote in line with the work of [Bo Pang 2002]. We took two publicly available lists of English words, one containing positive words and the other negative words. The list is the result of previous research about sentiment analysis on customer reviews and opinions on the web [Hu and Liu 2004] and [Bing Liu 2005].

The algorithm takes an individual review, counts the number of positive words (i.e. words that are on the positive word list), then count the number of negative words and compare them. If the sum of positive words is bigger than the number of negative words, the review is classified as positive; if the opposite happens, the review is classified as negative. If the number of positive words and negative words is equal is randomly assign them to positive or negative. (This is based on the fact that the distribution of ties is 51% positives, 49% negatives, based on the real values).

Bag of Words

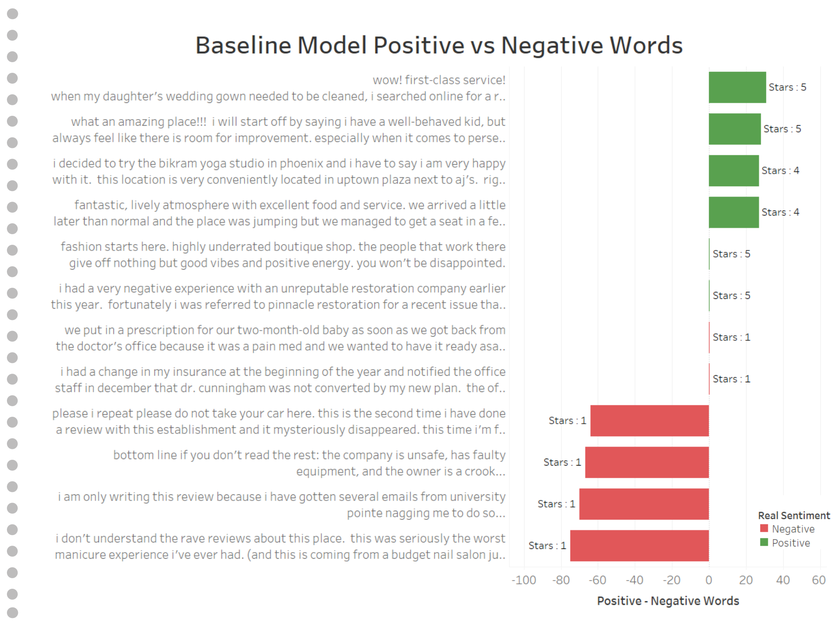
A bag of words model is a simplified representation of texts commonly used in Natural Language Processing. In this model, a text is represented as a multi-set of its words, disregarding its grammar and order but keeping multiplicity.
In this approach, the reviews were transformed into a document-word matrix. So, the (i,j)th cell of the matrix signifies the number of occurrences of word 'j' in document 'i'. The word list was arrived at after removal of Python's SciKitLearn's inbuilt list of stop words. Other filters applied were:

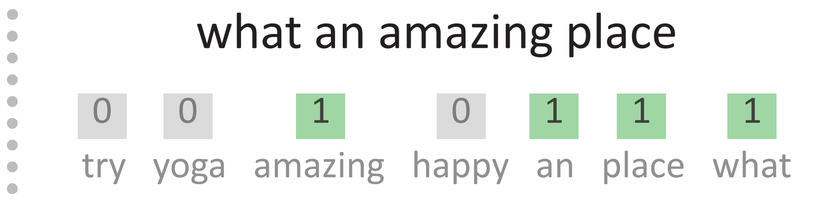
- The word had to exist in at least 100 documents in the corpus
- The word should not exist in more than 90% of the documents in the corpus

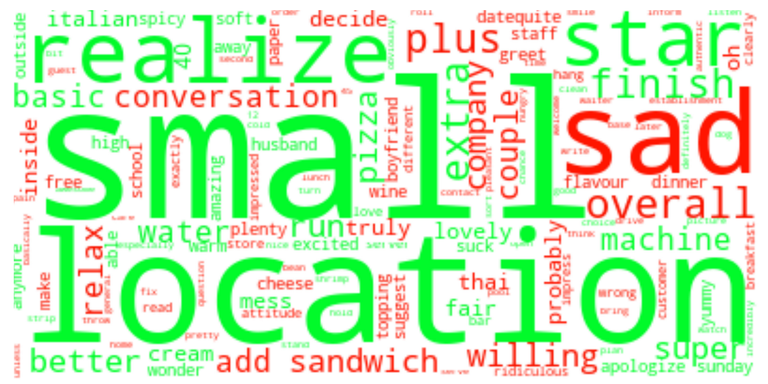
Pre-Trained Embeddings

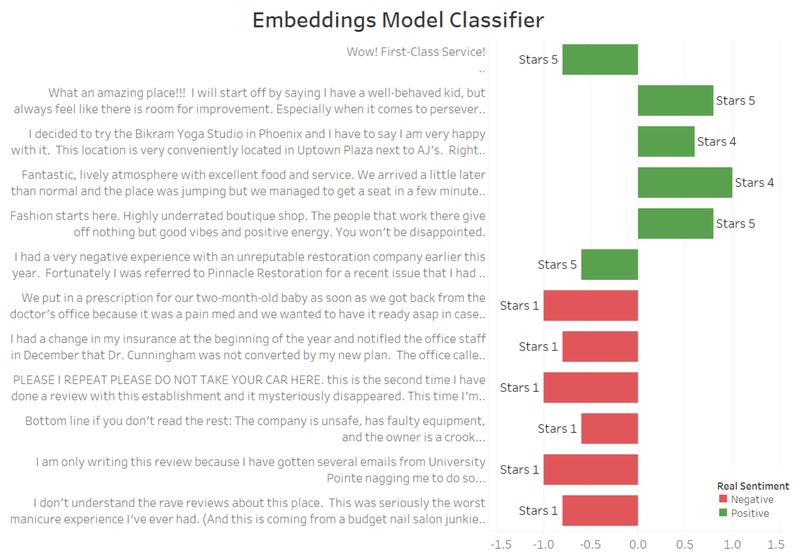
We use word embeddings as the starting point of these models—they are the building blocks of the feature matrix. A word embedding is a distributed vector representation of a word in a fixed-dimensional semantic space. The word embeddings are effective for measuring the semantic similarity of words [Pennington et al. 2014]. Pre-trained word embeddings have the advantage of being able to attach semantic meaning to a document with as few as a single word.

The way we construct the feature matrix is the following: we first delete stop-words and lemmatize words, then we attach a 200 dimensional GloVe embedding trained on Twitter reviews to every word, and we finally take the median of the words within each of the reviews over the entire dataset [De Boomet al. 2016].









RESULTS

Baseline Model

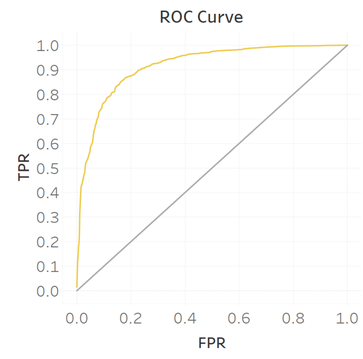
The baseline model classifies the words depending on the net count of positive words minus negative words. We took the positive and negative words from [Hu and Liu 2004] and [Bing Liu 2005]. Hence the baseline model doesn't need to train, and we tested it in the whole dataset. The range of positive words minus negative words is [-75, 32] with a mean of-0.51 and a median of 0. This baseline model has an accuracy of 0.76.

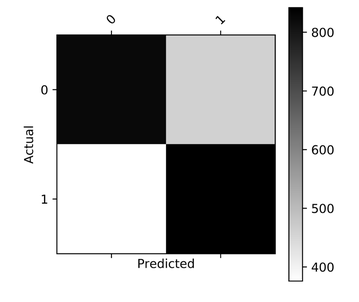
Bag of Words

Models shown below are fit on a feature matrix composed of frequencies of words present in each review. Each model is cross-validated on 10 folds of the training data. Scores denoted are accuracies corresponding to the hyper-parameter with the best accuracy score.

Pre-Trained Embeddings

Similar to the Bag of Words, the best cross-validation error for every model family is shown below. The table also shows the best models for every pre-trained embedding size.





Model Family	Cross-Validation Accuracy
SVM	0.873
Logistic Regression	0.871
Random Forest	0.858
Adaboost Forest	0.854

Model Family	Cross Validation Accuracy over Embeddings of d-dimensions			
	50	100	150	200
SVM	0.797	0.818	0.843	0.852
Logistic Regression	0.787	0.810	0.839	0.850
Random Forest	0.790	0.809	0.819	0.828
Adaboost Forest	0.788	0.799	0.819	0.837

Features	Test Accuracy
Baseline	0.760
BoW	0.851
PTE	0.845

REFERENCES

Junsheng Cheng Bing Liu, Minqing Hu. 2005. Opinion Observer: Analyzing and Comparing Opinions on the Web. Proceedings of the 14th International World Wide Web conference.

Shivakumar et al. Bo Pang. 2002. Thumbs-up?: Sentiment Classication using Machine Learning Techniques. Association for Computational Linguistics, The.

Gillian De Boom, Steven Van Canneyt, Thomas Demeester, and Bart. Dhoedt. 2016. . Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognition Letters* 80 (2016), 150–156.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. *To appear* (2017).

Minqing Hu and Bing Liu. 2004. Mining and Summarizing Customer Reviews. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Bing Liu. 2012. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies* 5, 1 (2012), 1–167.

Bing Liu and Lei Zhang. 2012. A survey of opinion mining and sentiment analysis. In *Mining text data*. Springer, 415–463.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, Vol. 10. 1320–1326.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 1532–1543.

Received April 2018; accepted —awaiting response