Predicting Company Ratings through Glassdoor Reviews

Fabian Frederik Frank
Stanford University
fabfrank@stanford.edu

Tyler Emerson Whittle
Stanford University
twhittle@stanford.edu

Mentor

Richard Socher

Problem Description

Reviews are a very common way of giving feedback to entities and helps the reader to get a sense of the quality. Such reviews are used in various contexts: For companies, books, films, etc. However, the qualitative nature of a review oftentimes leave an unsatisfying feeling of not being able to identify how to compare different entities just based on the review. A translation of a qualitative review into a quantitative metric can close this gap. Websites like Glassdoor.com already offer the option to also add a quantitative (star ratings out of 5) rating. The goal of this project is to predict those quantitative ratings by applying an advanced sentiment analysis to the qualitative reviews based on reviews and ratings from Glassdoor. Such a model can then potentially also be applied to review platforms where there is no quantitative component and therefore enrich the review and enable comparisons between different reviews.

Data

The data for this project comes from Glassdoor.com. We have written a python script that uses Glassdoor's API to scrape reviews from companies. Glassdoor.com has nearly 38 million reviews for more than 740,000 companies. A review contains the following sections: *title, pros, cons, recommends to friend, company outlook*, and *CEO approval*. Furthermore, it may contain information on: *advice to management, job title, years employed*. The data overall has 1.1 GB of text.

Evaluation Methodology

We are measuring the results of our model based on accuracy and an error term. For the accuracy, we care about whether we are getting the general modality correct. For that, we categorize a 4-star and 5-star rating as a positive rating, a 3-star rating as neutral, and a 2-star or 1-star rating as negative. The accuracy is the percentage of reviews where we get this general modality correct.

$$\texttt{Acc} = \frac{\#\text{of reviews where we predicted the right modality}}{\#\text{of reviews overall}}$$

In addition, we also calculate a precise error term, which is the mean squared error between the actual rating and the rating we predict.

$$\texttt{Error} = \frac{1}{N}\sum_{i}^{N}(rating_{predicted} - rating_{actual})^2$$

Baseline

Our baseline is training with x_i being a vector of binary values (1 if present, 0 if not) in review i regarding the 1000 most common words in our training dataset of reviews. y_i is the actual rating of review i (values

range from 1 to 5).
To create this vector, we first identify the 1000 most common words through a bag-of-words approach. To do this, we pre-process the reviews by stemming the words using the Porter stemming algorithm and by removing stop-words (like the, a etc.). We then convert each review into a vector of size (1000x1), with value 1 if the word from this index appeared (at least once) in the review, and 0 otherwise.
We tried out different training algorithms: logistic regression, negative binomial regression and a Naive Bayes classifier. The Naive Bayes classifier performed the best. The baseline tests will utilize a small subsample of the data consisting of 37,000 reviews.

Results

Classifying the reviews into positive, neutral, and negative categories using a Naive Bayes classifier yielded an overall accuracy of 63.4%. To further analyze the results, below we include the raw count and normalized confusion matrices. The normalized confusion matrix is given in parentheses below the raw count values. The classifier is best at classifying positive (4 and 5 star) reviews, while the accuracy is lowest for neutral (3 star) reviews. This is not unsurprising given there is a plurality of positive reviews in the data. We also extend the classification to predict the number of stars that a given review would have (1-5). Using this method, we can compute our error term defined in above. The calculation shows an error value of 1.22 for our sample data.

		Predicted		
		Neg	**Neu**	**Pos**
Actual	**Neg**	741 (0.53)	504 (0.36)	154 (0.11)
	Neu	223 (0.21)	455 (0.44)	367 (0.35)
	Pos	130 (0.06)	381 (0.16)	1856 (0.78)

Next Steps

Parameter tuning

As a first step, we will try to play around with various parameters to see whether those might have an impact on performance. For instance, instead of using a 1000-vector of words, we will try to use bigrams and trigrams to see whether the combination of words might be more powerful as a predictor. We will also consider varying the binary value in the vector to a count (e.g. if the word "good" appears 3 times, the value in the vector would be 3), to have some weighing of word appearances as well.

Sentiment analysis and deep learning

In addition, we will improve our model by using techniques from deep learning. [1] has shown that adding context vectors (CoVe) will improve performance of sentiment analysis. We plan to implement his architecture and apply it to this dataset.

References

[1] B. McCann, J. Bradbury, C. Xiong, and R. Socher. Learned in translation: Contextualized word vectors. In *Advances in Neural Information Processing Systems*, pages 6297–6308, 2017.