Predicting the Rating of a Company Using Employee Reviews

Aaron Marks
Department of Computer Science
Stanford University
aamarks@stanford.edu

Dhruv Kedia
Department of Computer Science
Stanford University
dkedia@stanford.edu

1 Introduction and Motivation

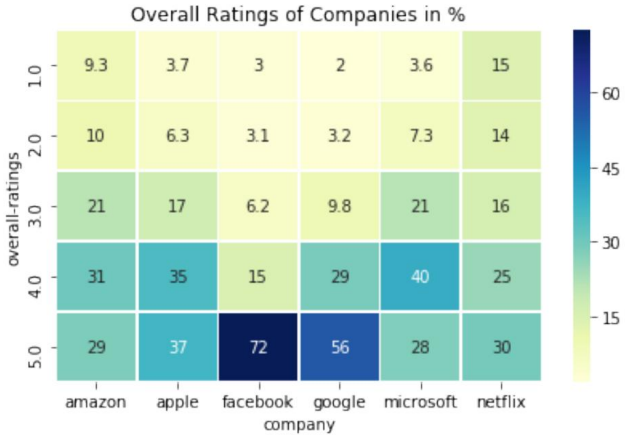
Using natural language processing on consumer reviews, whether they are movie reviews, Amazon reviews, workplace reviews is a common occurrence in studying sentiment analysis. This project seeks to use natural language processing techniques, including sentiment analysis, to predict employee happiness and satisfaction in the workplace. Many job seekers use the website Glassdoor to learn about employees' experiences at different companies before deciding to apply or accept job offers. Since the job seeker isn't asking an current or former employee is person what their experience was its important to be able to derive meaningful sentiment from the text reviews left on Glassdoor.

We are looking to explore different models to take reviews scrapped from Glassdoor and use NLP techniques to predict the employee satisfaction at a specific firm. Workplace text data is different from other types of sentiment data due to the industry specific and firm specific jargon used in reviews. One of the aspects of NLP we will be exploring vector space models in order to account for the specific jargon and word use. This will come in the form of using pre-trained word representations such as GloVe and Bert as well as building our own Word2Vec from our own dataset. In addition to this we will be experimenting with several deep learning models, both convolutional neural networks and recurrent neural networks. The input to are algorithm are company reviews by the employees and the output is the overall rating for the company.

2 Data

This data is scrapped from Glassdoor, an employee review site, and has been posted publicly on Kaggle. This dataset is composed of 67k employee reviews for Google, Amazon, Facebook, Apple, and Microsoft. Here is a heatmap showing the distribution of reviews of firms and their ratings:

Figure 1: Data Heatmap: Firm vs Overall Rating



These are all the associated values for each review: Company name, Location, Date Posted, Job-Title, Summary, Pros, Cons, Advice to Senior Management, Overall Rating (1-5), Work/Life Balance Rating (1-5), Culture and Values Rating (1-5), Career Opportunities Rating (1-5), Comp & Benefits Rating (1-5), Senior Management Rating (1-5).

The text data comes in four sections: Summary, Pros, Cons, and Advice to Management. We will begin by just using the summary for sentiment and will explore the other text sections later on. For the milestone, we are only looking to predict the Overall Rating.

In addition, we ensured that a random 95.0% of the data set was set aside for training, 2.5% for validation and 2.5% for test. The reason we chose to use an this split is because our dataset is rela-

tively large and around 1650 examples are enough for validation and testing. The rest of the examples will be more helpful for the model to learn and train.

We also took some steps to pre-process the data. To begin with we removed all comments that did not have a summary or the summer value was "NaN". Then for all the comments we removed punctuation, new lines, and stop words like "a", "and", "the", etc. Additionally, we also made all the comments lower case.

3 Method

3.1 Model using Summary Comments

To begin with we only take into account the summary comments reviews provided by employees. We have chosen to ignore the pros and cons reviews provided by employees. This is because initially we believe summary comments will help provide the overall sentiment for the employees experience at the company.

3.1.1 Word Embeddings

The first step we did was create word-level embeddings for the different words observed in the summary comments by employees. We created these word-level embeddings by using 50D GloVe encodings that used 6 Billion token pre-trained embeddings. These word embeddings were then concatenated to form sentence level embeddings. We had a maximum length for each sentence to ensure that the sentences embeddings had same dimensions. If the maximum length wasn't met, the sentence was padded with "pad" tokens. A potential problem of using GloVe embeddings is that some of the words appearing in reviews might not be present in the GloVe embeddings. For example, technical jargon specific to the companies and colloquial or slang words. In order to address this problem, we experimented with two approaches: 1) We ignored the word completely and moved on to the next word. 2) An UNK token is used whenever an unfamiliar word is encountered. The UNK vector is set randomly with the same mean and variance as the rest of the embedding space..

3.1.2 Logistic Regression

The logistic regression model makes use of the logistic or sigmoid function which is as follows:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{1}$$

where theta are the feature weights and x is the input example.In multi-class logistic regression, we made use the One vs Rest (OVR) model. The OVR model trains a logistic regression classifier $h_\theta^{(i)}(x)$ for each genre "i" to predict the probability that $y = i$. For each new input "x", the genre "i" that maximizes $h_\theta^{(i)}(x)$ is the predicted genre.

For the baseline model we simply implemented the OVR model because it allows for experimentation and ease of understanding. However, a disadvantage of training a separate binary logistic regression model for each genre is that it assumes the probabilities for each class are independent.

3.1.3 Convolutional Neural Network

For a baseline deep learning model we used a simple convolution neural network architecture. A CNN is similar to a simple neural net, but replaces most of the fully connected layers with convolution layers. The convolution layers parameters consist of a set of learnable filters. Each filter is small spatially, and as we slide the filter over the width and height of the input volume, we will produce a 2-dimensional activation map that gives the responses of that filter at every spatial position.

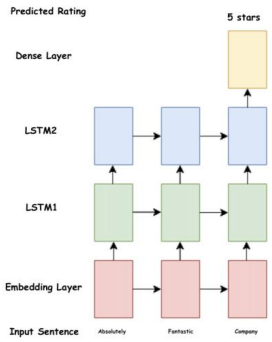
The CNN architecture we used is as follows: An embedding layer. Followed by 2 Conv1D layers with 64 filters, 5 kernel size, stride 1, no padding and a ReLu activation function. Followed by a MaxPooling1D layer with pooling size of 2. Followed by a linear layer with 1000 units and a ReLu activation function. And lastly, an output softmax layer since it is a multiclass problem. Additionally we used the categorical cross entropy loss function and an Adam optimizer.

3.1.4 Recurrent Neural Network

RNNs are designed to work better on sequential data: when the information at one time step is related to the information at some previous time step. This makes them ideal for use on text data and a problem like sentiment classification. This is because text data is sequential, what comes next in a sentence is often conditioned on what came before in the sentence. Hence, it is important to use a model that exploits sequential information. RNNs allow the network to remember the input state that appeared previously to make predictions about the future. However, a problem of RNNs is that as the sequence grows large, however, the gradients can become infinitesimally small. Hence, in our model

we make use of LSTMs to tackle longer reviews. LSTMs develops a memory cell and uses gates to decide how much information needs to be forgotten or need to flow through the time steps. In this way, useful information can be kept and unnecessary information can be dropped. (9) The model we use is as follows:
1) LSTM layer with 128 neurons, 0.2 dropout
2) LSTM layer with 128 neurons, 0.2 dropout
3) Dense layer with softmax activation
The visual representation of the model can be seen in figure 2. Further on this model uses a categorical cross entropy loss function, an Adam optimizer, and tries to optimize for accuracy.

Figure 2: RNN Architecture



3.2 Model Using Summary Comments, Pros, and Cons

One aspect we noticed was that summary comments will not always be able to grab the views and feelings of the employee. For example, if an employee was very dissatisfied working at a particular company and rated the company a 1, their comment for cons will be more detailed and provide more insight as to why. Hence, it will help in predicting that the rating was a 1. Similarly in the case of an employee being very satisfied with a company, the pros might prove to be extremely indicative of the rating. In order to test this hypothesis we decide to experiment with using pros and cons comments along with the summary comments.

3.2.1 Concatenating All Comments Together

As a first step we concatenated all the 3 reviews to be part of 1 review. We then calculated the sentence level embeddings as described in section 3.1.1 and ran the RNN based model described in section 3.1.4. In hindsight this was not the best idea as concatenating all these 3 reviews would lead to 1 very review with a very confusing sentiment. Hence, this would make it difficult for the model to learn and perform well.

3.2.2 Training Individual Models

Initially we maintained 3 different sentence level embeddings calculated as described in section 3.1.1. We then ran an individual RNN based model as described in section 3.1.4 on each of the different kind of reviews. We then used the outputs of the three different RNN LSTMs, and added a fully connected dense layer. This had a softmax activation function that produced predicted probabilities for each of the possible rating buckets.

4 Results

4.1 Evaluation Metrics

In order to evaluate individual model performance, we used the following metrics:

$$Top1\ Accuracy = \frac{Correctly\ Classified\ Examples}{All\ Examples} \tag{2}$$

$$Top2\ Accuracy = \frac{Correctly\ Classified\ Examples}{All\ Examples} + \frac{Incorrectly\ Classified\ 1Degree\ Away}{All\ Examples} \tag{3}$$

Example of to 2 accuracy is as follows: if the actual rating is a 4 but it is mistakenly classified as 3 or 5, then in the case of top 2 accuracy, it is considered accurate.

For the best model, we also plotted the loss curves and the accuracy curves over epochs.

4.2 Baseline Model Results

Table 1: Summary of Results

Model	Train Accuracy	Val Accuracy
LR	0.36	0.34
CNN	0.46	0.46

From the initial results we can see that both the Logistic Regression and CNN model failed to perform well. Here we can see that the training and validation results are pretty similar. We can clearly see that the model is not over fitting the training dataset well and is hence not able to build

a complex enough model that is able to learn well. This is a case of a high bias model. However, the CNN model did outperform the Logistic Regression model. This is probably due to it being a deep learning model. This being said we had not tuned any of the hyper parameters of the model and had only ran the model for 5 epochs.

4.3 RNN Model Results

Training Top 1 Accuracy = TT1A
Validation Top 1 Accuracy = VT1A
Validation Top 2 Accuracy = VT2A

Table 2: Summary of Results

Model	TT1A	VT1A	VT2A
Summary Only	0.62	0.58	0.77
All Reviews Concat	0.42	0.41	0.64
All Reviews Individual	0.78	0.62	0.88

Here we can see that the RNN model outperforms the baseline models. This is expected as we described above RNNs perform better on text and sequential data. From the models we can also see that concatenating all the reviews does not lead to performance improvement. However, building individual models for each type of review and adding a softmax layer leads to a significant improvement in performance. This helps satisfy our earlier hypothesis that cons will help more in the classification of bad reviews and pros will help more in the classification of good reviews.

Beyond this, it is also interesting to note the different in top 1 and top 2 accuracy performance. In every case we can see that the model performs significant better if we use the top 2 metric for comparison. This makes sense because it is effectively similar to narrowing the number of classes from say 5 to 3. Also, on the other hand many times it is extremely difficult to distinguish between a 4 star and a 5 star review or a 1 star and 2 star review. This is even true in the case of humans. The rankings in terms of performance for these different kind of models doesn't change when comparing using top 1 or top 2 accuracy.

4.4 Best Model Results

The best model is the one where 3 individual RNN models are trained for summary comment, pros comment, and cons comments, and then finally combined with softmax layer. The model uses pre-trained GloVe word embeddings, dropout of 0.2, learning rate of 0.2 and an Adam optimizer. The below graphs plot the accuracy and loss for this model.

Figure 3: Loss vs Epochs

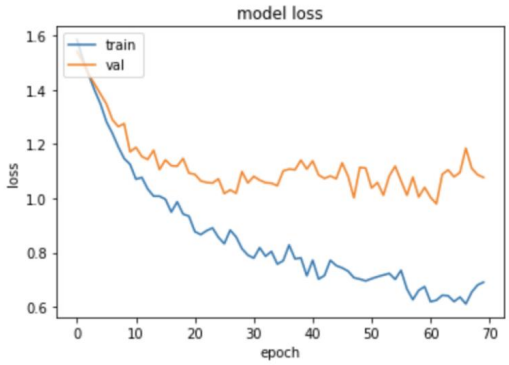
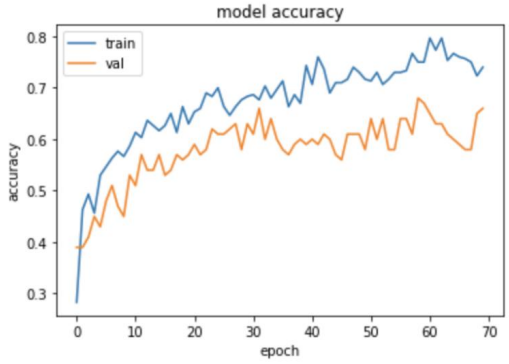


Figure 4: Accuracy vs Epochs



Given that this was the best model, we evaluated the results on the test. We managed to achieve a test top 1 accuracy of 0.63 and test top 2 accuracy of 0.86.

5 Qualitative Evaluation

In order to identify words that were strong classifiers of 5 star and 1 star reviews, we ran a count script to explore which words appear most in these reviews and our model is successfully able to classify. We observed that words such as "great", "best", "smart", and "friendly" our model was successfully able to catch and rate as either 4 or 5 stars. On the other hand words such as "poor", "stress", "burnout", "bad" were words that our model was successfully able to catch and rate as either 1 or 2 stars.

On the other hand, it was also extremely interesting to see words that the model often failed

to catch. Words such as "unique", "different", "exciting", and "new" were some top words associated with 4 or 5 star reviews that the model failed to catch successfully many times. Words such as "long", "boring", and "free" were some top words associated with 1 or 2 star reviews that the model failed to catch successfully many times. Also there were certain phrases that were ambiguous and led to poor classifications, for example: work life balance, technical understanding, and company culture. This is due to humans not writing reviews in full sentences and just leaving them as bullets in both pros and cons sections.

Regardless of improvements, it's clear that the signal from 5 star reviews is strong and from 1 star reviews is strong however, like many multiclass problems the signal becomes muddle in the classes between the extremes. We discovered that there is a high human error rate for this problem and dataset by predicting ratings ourselves. Many employees leave reviews with sentiment mismatching the ratings.

6 Conclusion

As we have seen above the best model uses 3 different RNN based models with GloVe word embeddings. However, even this best model has a lot of scope to learn more and overfit the training data. A deeper model or a different architecture can go a long way in improving the model. Also we can see that the model performs significantly better when using top 2 accuracy as opposed to top 1 accuracy. Our limitations working on this project was training time. It prevented us from further hyper-tuning several of the parameters.

7 Next Steps

In terms of next steps we want to address three main areas: data usage, word embeddings, and the model itself.

Firstly we could take more steps to preprocess the data. For example, there are several comments such as typos, random characters, or simply empty comments. It would helpful to remove these examples as they are not useful for the model to train on.

Next as mentioned above, a problem with the GloVe word embeddings might be that some of the words appearing in reviews might not be present in the GloVe embeddings. We would like to experiment with using Bert and Elmo word embeddings. Additionally we would also want to experiment creating and training our own word embeddings using the word2vecmodel and a neural network architecture.

Our model does not seem to be performing extremely well. It still doesn't learn perfectly. Hence, we will experiment with a deeper recurrent neural network, train it for longer number of epochs, and hypertune the parameters. This is especially true in the case where we are training individual rnns for each type of comment. The model took very long to run so we did not have as many resources to hypertune the parameters. Our RNN architecture is one we built on our own, it might be worth exploring the state of art RNN architecture on a sentiment analysis task such as the Stanford Sentiment Twitter dataset. Lastly, since we are using multiple RNNs it might be useful to explore using attention networks. Attention networks are currently state of the art in this field and could help improve our results significantly.

Acknowledgments

We would like to thank Professor Kian Katanforoosh and Professor Andrew Ng for all of their help and guidance with the project. We would also like to thank Weini Yu for all her mentorship and great suggestions along the way.

Contributions

Aaron and Dhruv contributed equally to the project. We built the models together in person. Dhruv focused more on writing of the paper and Aaron focused more on creating the poster.

References

[1] Amit Schechter, Camille Pataki, Dhruv Kedia, CS231N Final Paper.

[2] Dhruv Kedia, Swathi Iyer, Vidushi Singhi, CS229 Final Paper.

[3] Dongxu Zhang, Dong Wang. Relation Classification via Recurrent Neural Network. (2015) *http://arxiv.org/abs/1508.01006*

[4] Dos Santos, Cicero & Gatti de Bayser, Maira. (2014). Deep Convolutional Neural Networks for Sentiment Analysis of Short Texts.

[5] Google, Amazon and more Employee Reviews, Kaggle. *https://www.kaggle.com/petersunga/google-amazon-facebook-employee-reviews*

[6] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation.

[7] . Journal of Machine Learning Research, Vol 12, pgs 2825 - 2830, 2011.

[8] Karpathy, Andrej. "Convolutional layer" CS231N Lecture Notes.

[9] Ng, Andrew. "Recurrent Neural Networks" CS230 Lecture Notes.