"My wife and I hate this place"
A guide on writing humourous Yelp Reviews as learned through n-gram features

Richard Gao
A53073831

December 1, 2015

CSE255 Assignment 2

I. Summary

In the following study, I attempt to model funniness as a function of linguistic features. The dataset consists of a subset of 1.6 million Yelp business reviews, each of which can be voted as "funny" by review readers. I design a normalized funniness rating, or F Index, and predict a review's hilarity with n-gram features learned through linear models. The model performs better than baseline, and I observe that the funniest reviews are apparently those involving terrible businesses, bad owners, and horrific dining experiences suffered with one's wife.

II. Dataset Exploration

The dataset in question is a comprehensive collection of Yelp review data, provided by the Yelp Dataset Challenge [ref], gathered from Yelp mobile and web app where users rate their experience at restaurants and other businesses. The dataset includes 1.6 million individual reviews (user-business pair, review text, star rating, and "votes" for the review), spanning 60,785 businesses and 366,715 users.

Review Voting

Similar to Amazon's review helpfulness rating, each review can be voted as useful, funny, cool, or any of the combinations thereof. Unlike Amazon, however, the rating interface does not allow negative votes (i.e., not useful), creating an implicit response bias.

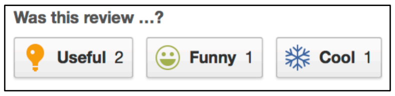

Fig.1 Yelp review rating interface

The reviews for a particular business are by default displayed in an order based on "Yelp Sort", which likely takes into consideration of various factors such as user trustworthiness ("Yelp Elite"), review votes, recentness, etc. Therefore, we would expect to see a few reviews with many votes and many reviews with little or no votes, which is exactly what we observe in the histogram data, where the straight line in log-log plot demonstrates an inverse power law (Fig.2). In addition, the average of the 3 categories for follows a similar trend. Interestingly, we see that reviews are more likely to be voted as funny than as cool or useful, which is indicated by the total categorical votes (cool: 751,499, funny: 1,681,655, useful: 932,429).

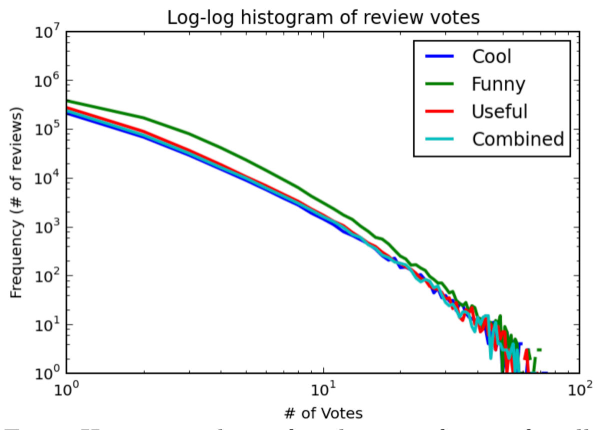

Fig. 2 Histogram data of each type of votes for all reviews, plotted in log-log scale.

A power law distribution does not necessarily imply that the rich gets richer, as it is possible that only a few reviews are truly useful, funny, or cool. However, when we observe the pairwise correlations between the 3 rating categories, we see that they are strongly correlated, suggesting that the underlying factor leading to high ratings to any and all 3 of the categories is simply to gain momentum in popularity. To control for artificially high correlations related to many 0-rating reviews, reviews with less than 1 (and 3) total votes coming from any categories were excluded (834,146 & 392,917 reviews remained), but strong correlations were still present.

	C-F	C-U	F-U
All reviews	0.732	0.799	0.836
1 or more votes	0.707	0.778	0.815
3 or more votes	0.681	0.763	0.809

Table. 1 Pairwise correlation coefficients. C – cool; F – funny; U - useful

Corpus Analysis

Building a corpus of words from the entire database yields 804,626 unique words (unstemmed), 804,499 of which remain after removing English stopwords. The frequency of occurrence of words follows an inverse power law, as expected, and the most common are stopwords (e.g., 'the', 'and', 'I', etc). Bigrams follow a similar pattern, though there are many more bigrams (12.5M) than unigrams.

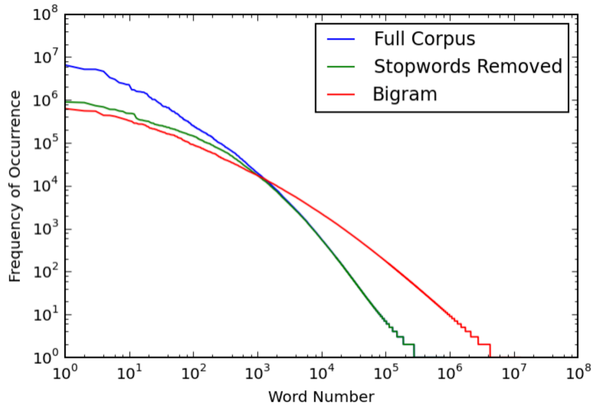

Fig. 3. 1- & 2-gram frequency

III. Predictive Task

The high-level goal of this exercise is to learn, from the Yelp review dataset, what makes a review funny, based on its linguistic features alone. In another framing: how can we help an user write a funny review to the best of their capabilities, given that they cannot change factors such as their popularity in the short-term? To decide whether any model has captured a notion of what's funny, we measure how accurately it predicts "funniness" of unseen data. A subtle but important point must be noted here: the goal is not to predict funniness as accurately as possible, per se, but to do so using only information from the review text.

As shown in the previous section, the funniness vote is heavily dependent on various non-linguistic factors, such as popularity, reviewer exposure, reviewer history, etc. Two approaches are possible in dealing with this issue: the first is to include all possibly relevant information as input (review date, location, etc.) and simply use the funniness vote as the output, relying on the model to tease out different dependence structures. In other words, we could look at how much a good linguistic model improves prediction accuracy, given all other relevant information.

The second approach is to limit the input to only text features, and design an output metric that already accounts for the effects of non-text features. Here, I take the latter approach – at the cost of a possibly biased output label – for a few reasons. First, the funniness vote scales almost linearly with total votes number (i.e., exposure), and regressing for it would answer an interesting, but different, question, namely: what makes a review popular. Secondly, with a bit of human intuition, we should be able to come up with a relatively independent measure of funniness, as I outline below. Finally, having comparable feature types (e.g., word and phrase occurrences) makes the model easier to interpret in the end, as we can directly compare feature weights to judge what's commonly thought of as funny.

<u>Feature & Output Design</u>

Given that the funny vote scales linearly with total number of votes, we would like to use a normalized measure as output instead. I first explore the effects of normalizing the categorical vote counts by either the total votes of all 3 categories, or the maximum vote. Normalizing by sum caused categorical votes of popular reviews to converge to 1/3, effectively suppressing variance for samples where differences could be most informative, making it an undesirable option (Fig. 4). Normalizing by max created a more independent measure from 0 to 1, though, as noted earlier, the total number of funny votes is greater than either of the other two, thus creating a disproportionate number of samples with a funniness of 1.

Considering the interface again, we see that one can cast a vote in any or all three categories. Thus, if popularity alone were driving all 3 votes, they would grow at roughly equal rates. However, if a reader consciously casts a vote in funny but not in the others, then we can deem a review to be funny, or at least more funny than cool/useful. Therefore, I define the funniness index (F Index) as how many extra funny votes a review received compared to the mean, i.e.

$$F\ Index = funny - \frac{(cool + funny + useful)}{3}$$

The F Index is distributed symmetrically about 1 (Fig. 5). Interestingly, reviews with more total votes tend not to have extreme values for F index. This is counter-intuitive as the more votes a review receives, the larger a margin can exist, which is seen in the fan-out pattern when total votes is below 50. One possible weakness for this measure is that it does not only measure how funny a review is, but how not useful it is, comparatively.

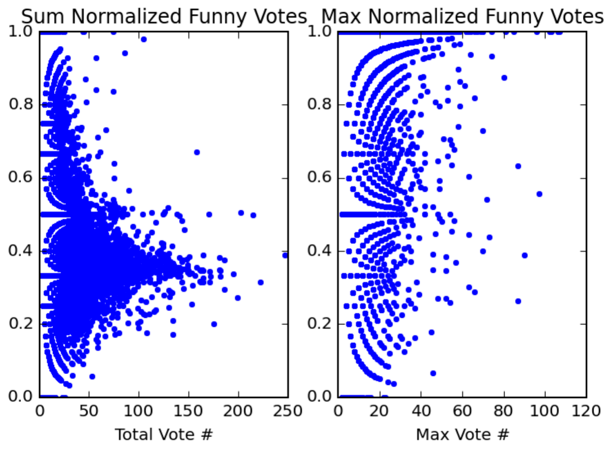

Fig. 4 Funniness ratio, as normalized by total votes (left) and max votes (right)

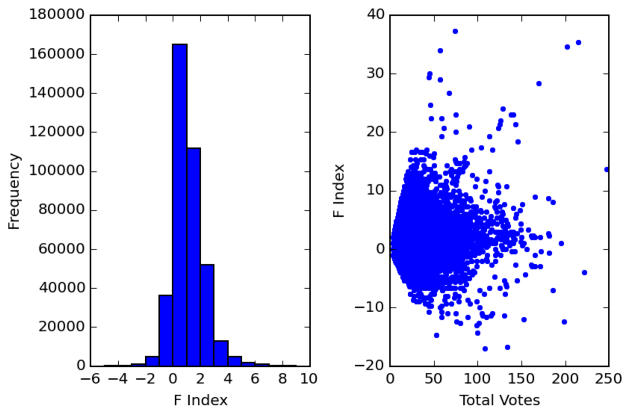

Fig. 5 Distribution of F Index (left) and F Index as a function of sum of votes (right)

Having achieved a satisfactory output measure, the input features are relatively easy to build. For this task, I focus on text features of the reviews, building a corpus of uni-grams and bi-grams using the entire dataset of 1.6M reviews. I explore the effect of using different numbers of dimensions, as well as representing the input features in different ways, comparing raw counts to weighted counts (tf-idf). Obviously, language has sequential structure and syntax, as well as contextual information, which is why a phrase like "I can't even" might be funny in some cases and perfectly vanilla in others. As such, an input representation that retains structure is ideal. Without using a complex model that generalizes over sentences to obtain structures such as noun-verb-noun, n-grams provide a good balance between keeping low dimensionality while retaining some semblance of structure. The task, then, is to predict a review's F Index using its text features, while the underlying goal is to look at which words and phrases are weighted most heavily in the best performing models.

IV. Model Selection & Experiments

For learning, I use only reviews with 3 or more votes (392,917 samples) in total in order to ensure proper sampling and calculation of F Index. I use regression models (linear, ridge, lasso, and elastic net) to map input to output, because they are the easiest and most natural models to use given the problem I defined. More sophisticated models could be applied, such as artificial neural networks, but they come at a cost of model interpretability, which is important for translating model parameters to our intuitive notion of funniness. To evaluate model performance, I calculate the mean squared error (MSE) of the predicted F Index on the unseen data. Since there are no real test sets, I evaluate the accuracy of each model via randomized 4-fold validation, comparing the mean MSE on the rotating validation set for each model. The baseline of this task is set with the bias-only model, i.e., predict every validation sample as the mean F Index of the training set.

For the following report, I perform 3 main experiments to optimize the prediction error. First, I vary the dimensionality of the input representation, taking the N most common word features and assess whether accuracy scales with dimensionality. Second, I use several different input representations, including unigram, bigram, a mixture of both, as well as TF-IDF representations for all the above. Finally, I (concurrently) investigate the effect of different regularization schemes on prediction accuracy, to see whether overfitting of ordinary least squares is an issue.

V. Result

Input Dimensionality

Does prediction accuracy improve by using more words? For the first experiment, I vary input dimensionality with several different models. The input consists of the top N most frequent unigrams, where N = [50, 100, 500, 1,000, 2,000]. The baseline performance is set using the mean model, i.e., always predicting the mean F Index of the training set, which achieves an MSE of 1.58. Fig. 6 summarizes the main findings. I find that MSE decreases monotonically as a function of dimensionality, up to 2,000, which is the max value my machine is able to run in a reasonable amount of time. This is reasonable as there are 800k unique unigrams, so the first 2,000 are likely to be common to almost all reviews, thus preventing overfitting. In addition, I compare the performance of regular linear regression with ridge regression of various regularization parameters, and find that there is no practical difference between these models. Again, this is likely due to the fact that I have no crossed into a high enough dimensionality for overfitting, and both the input and the output are nicely distributed with no extreme outliers. As such, all models for the following experiments are trained on 2,000 features.

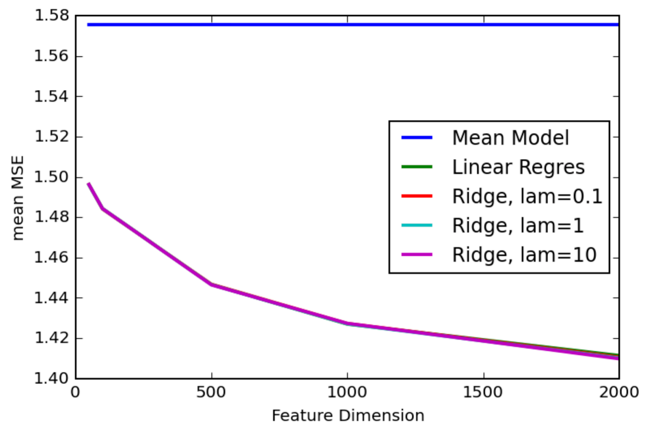

Fig. 6 MSE for various input dimensionality and regularization

Input Representation

Do individual words capture funniness better than phrases? And do rare words evoke a funnier reaction? I investigate this question by using different representations of the input. By keeping the number of dimensions the same, I use the most common 2,000 unigrams, bigrams, or an equal mixture of the two as input. As shown in Fig.7, unigrams alone provide the lowest MSE, followed by a mixture of the top 1,000 most common unigrams and bigrams, and bigrams alone perform the worst. To see whether the most common words were the contributing factor, I use the 1,000th to 2,000th most common unigrams and bigrams (last column) instead of the most popular 1,000, and performance was much worse. Therefore, combined with the results from the previous experiment, it seems that most information is contained in the relatively common words, and adding more words improves performance.

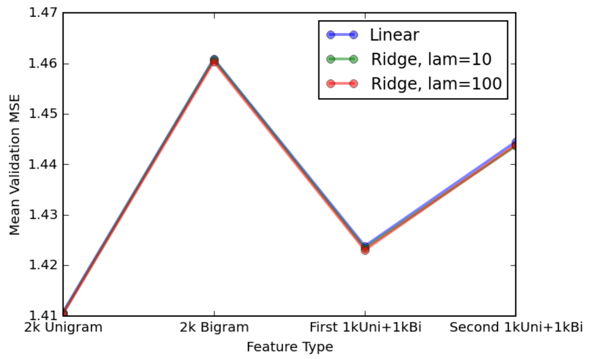

Fig. 7 MSE of various models using unigram, bigram, and a mixture

To account for the effect of review length (i.e., using more common words more often), I transform the features using TF-IDF weighting for both the unigrams and bigrams, and perform the exact same experiment as above. TF-IDF is calculated as shown in lecture and various other sources, where TF is the raw count, and IDF is the logarithm of the ratio between total document count divided by documents containing a particular term, and TF-IDF is simply taken as the product of the two. In addition to regular and ridge regression models, I also included lasso and elastic net models. Fig. 8 summarizes the main results.

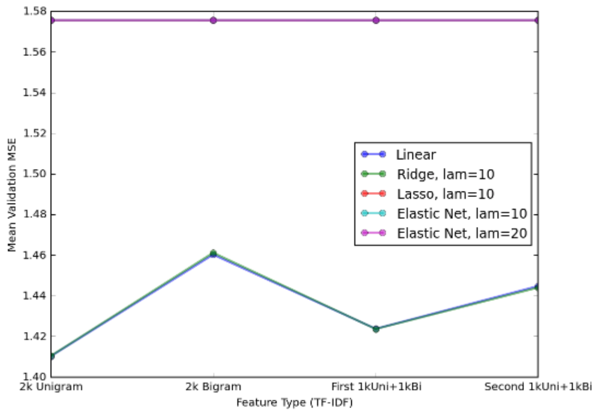

Fig. 8 MSE of various models using TF-IDF representation. Linear and ridge regression traces are stacked on the bottom, and the other three are stacked near the top.

Prediction accuracy using TF-IDF features are almost identical to that of using raw count features, which is surprising. Even more surprisingly, the most heavily weighted features in both sets of models are extremely similar, i.e., the same words/phrases were weighted heavily regardless of whether raw counts or weighted counts were used. A possible explanation for this is that the most discerning words and phrases have roughly the same document frequency, and the non-informative words occur with relatively constant frequency in all the documents.

<u>Regularization</u>

Alluded to in the above experiments already, regularization did not seem critical in this particular situation. Note that in Fig. 8, only linear and ridge regression performed above baseline, whereas lasso and elastic net failed miserably. I'm not sure why this happened, as those 3 models were re-run with smaller regularization (1.0) and the same result occurred. It's possible that the default maximum number of steps in lasso and elastic net optimization was reached before converging to a satisfactory solution. It was also interesting that ridge regression trained much faster than linear regression, probably due to the exact but slow matrix inversion in the regression implementation.

VI. How To Be Funny on Yelp

Fig. 9 (final page) shows the 25 highest weighted (positive and negative, excluding offset) phrases from the combined unigram+bigram linear regression model. A few interesting observations can be made. First, it seems that being polarizing makes for funny reviews, particularly with a leaning towards negative sentiment ([rude, will never, worst, terrible, horrible] compared to [incredible, excited]). Conversely, the most negatively weighted phrases are relatively neutral. Secondly, 'my wife' and 'wife and' are both funny terms, whereas 'wife' by itself seems to be the opposite (same with 'boyfriend' vs. 'my boyfriend'). This is a rather strange effect, possibly explained by the fact that the mentioning of 'wife', and therefore any bigrams including 'wife', makes a review not funny, but specific mentions of the reviewer and his wife somehow counterbalances this effect. Finally, it appears that mentioning of a third-person noun ('man', 'girl', 'guy') is associated with a rather unfunny review. These interpretations are, of course, correlational, and do not take into consideration of the interaction between the terms. And as previously mentioned, other unaccounted effects, such as popularity, trustworthiness, etc., may still remain. However, it does provide some insight as to when Yelp reviews are rated as particularly funny.

Based on the first observation, I explored whether review rating was correlated with F-Index, which, admittedly, should have been one of the earliest exploratory analysis. Indeed, it appears that both 1- and 5-star reviews were more likely rated as more funny, which corroborates with the previous observation regarding polarized phrases (1-star more so, see Fig. 10). When the star rating (and star-squared) was included in the same 2,000-dimensional input, performance improved to 1.395 with the unigram model and 1.406 with the combined unigram-bigram model, better than all previous models. Additionally, regressing for the star ratings almost completely removed the negative phrases, while keeping most of the others top ones intact (Fig. 11).

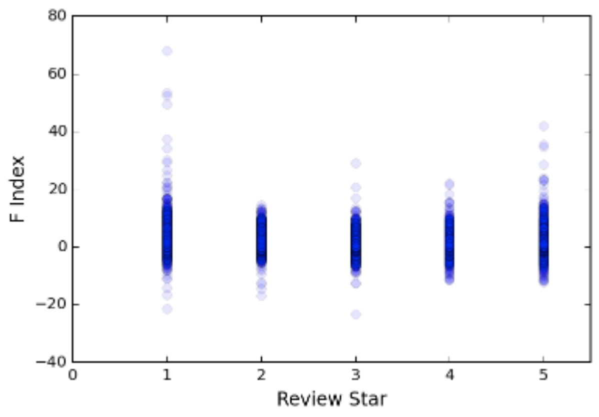

Fig. 10 F Index as a function of star rating

VII. Conclusion

In summary, using linear models and linguistic features, I learned that to write a funny Yelp review, you should bring your wife to a bad restaurant. Interestingly, previous studies on linguistic models of humor have reflected this observation. In Mihalcea & Pulman (2007) [2], they used humorous and serious news texts (the Onion vs. BBC) as input and produced binary classification using logistic regression, SVM, and naïve Bayes. They found that humorous articles tend to be overwhelmingly negatively polarized, as I have replicated here. In addition, they found that humorous articles were also more human-centered, which I was not able to reproduce. In that study, they achieve 96% classification accuracy on news articles using SVM. An extension of my current study would be to categorize reviews based on their F Index into funny vs. not funny, or funny, neutral, not funny, and explore which linguistic features are weighted heavily in a categorical classification. The advantage of doing such a task would be to ignore the magnitude of funniness, as I have currently included with a continuous F Index. Buscaldi & Rosso (2007) [3] attempts a similar task using n-gram representations of Italian quotations, classified using Bayesian classifiers and SVM. In general, humor classification is similar in nature to SPAM classification, and categorical classifiers such as Bayesian and SVM models are commonly applied on bag of n-gram inputs. Although categorical classification is superior in some cases, the current dataset presents an excellent opportunity to assess humor along a continuous dimension, though one may argue that how many people find it funny does not equate to how funny it is.

The current analysis can be improved in several ways. As mentioned above, it would be interesting to see which features become important when a discrete measure of funniness is used. In addition, I specifically ignored other features present in the Yelp dataset in order to form a constrained and easily interpretable task, but that does not have to be so. Previous work on this dataset has been focused on building recommender systems, or assessing linguistic valence, but not humor in particular, though complex models that take into account of geographical location, time, average rating of the business under review, and use attributes can certainly be applied in this case [1]. Finally, as I already discussed in the opening paragraphs, more complex models, such as recurrent neural networks, are well suited to capture the sequential structure of language, which could provide further insight compared to a limited representation using uni- and bigrams.

VII. References

[1] Yelp Dataset Challenge. http://www.yelp.com/dataset˙challenge

[2] Mihalcea, R., Pulman, S. 2007. Characterizing Humour: An Exploration of Features in Humourous Texts. https://www.cs.ox.ac.uk/files/244/mihalcea.cicling07.pdf

[3] Buscaldi, D., Rosso, P. 2007. Some Experiments in Humour Recognition Using the Italian Wikiquote Collection. Chapter in WILF 2007, LNAI 4578, pp. 464-468. http://users.dsic.upv.es/˜prosso/resources/BuscaldiRosso˙CLIP07.pdf

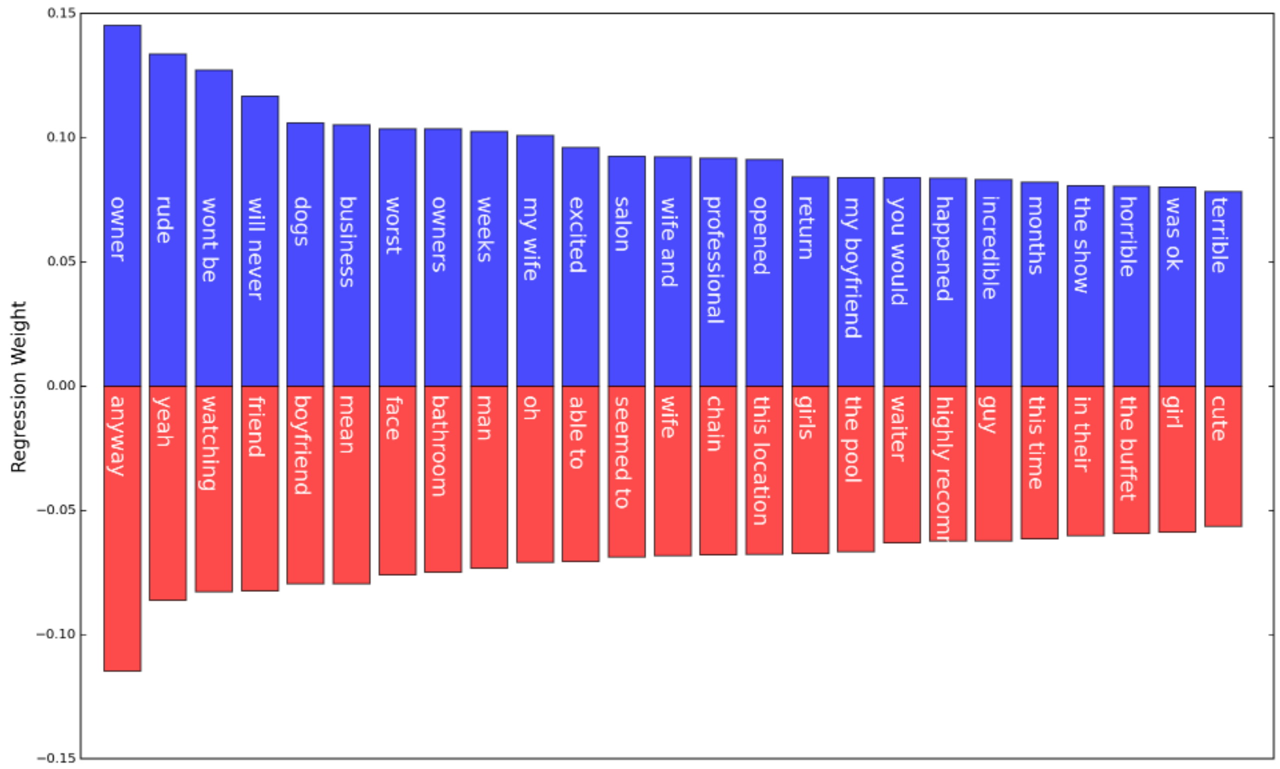

Fig. 9 Top 25 most associative unigrams and bigrams

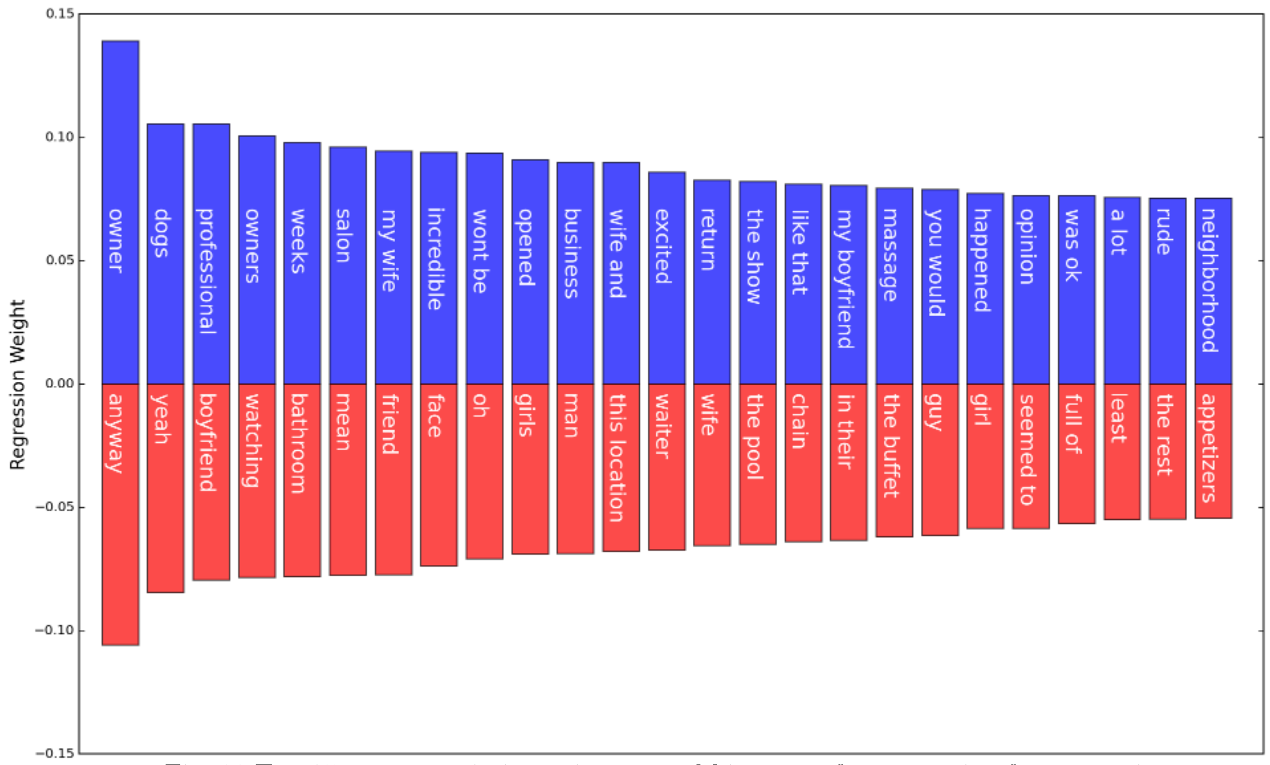

Fig. 11 Top 25 most associative unigrams and bigrams, after regressing for star rating