COMP30027 Machine Learning
Project 2: *'This Review Sounds Positive'*

1 Introduction

Sentiment analysis is the computational extraction and classification of subjective material from concrete media. It is invaluable to businesses for voice of the customer (VoC) initiatives such as review mining, and customer profiling applications including recommender systems and targeted advertising. One of the most basic applications of sentiment analysis is polarity analysis; in this report, three models are presented to classify polarity by predicting the star ratings of Yelp text reviews[1].

2 Preprocessing

In natural language processing, some variant of word embedding is almost always necessary so that textual input may be understood by computers. Two types of word embeddings have been considered, a Common Bag of Words representation and three Doc2Vec representations. Baseline scores for these formats generated with five fold cross validation of untuned random forest classifiers are given in Table 1.

CBOW	Doc2Vec		
	50	100	200
75.10%	77.13%	74.65%	72.11%
75.90%	77.56%	74.88%	72.48%
76.52%	77.97%	76.02%	73.89%
75.18%	77.73%	75.20%	72.58%
75.84%	77.68%	74.90%	72.46%
75.71%	**77.61%**	**75.13%**	**72.70%**

Table 1: Word embedding comparison of RFC baselines with 5-fold CV and average accuracies

The accuracy of the models trained on Doc2Vec embeddings was found to be inversely correlated to the dimensionality, likely due to the models being unable to fit the increasingly more complicated information captured; however, more powerful models could yield better results on the higher dimensional representations. Between the two word embedding techniques, the bag of words representation actually performed roughly on par with the average Doc2Vec model. This is possibly due to CBOW's comparatively simplistic information representation. However, Doc2Vec is known to generally perform better, as locality and contextual information is lost in CBOW, and additionally Doc2Vec also encodes actual semantics[2]; the fact that Doc2Vec 50 significantly outperformed CBOW is testament to this. CBOW's sparse matrix representation also makes it more difficult to feature engineer, additionally being incompatible with some of scikit-learn's built-in classifiers. Hence it was decided to use a concatenation of all three Doc2Vec embeddings; while this might initially introduce data redundancy, performing feature selection should ultimately be able to yield the maximal performance in this way. Not very true, there's always the option of stemming and lemmatising

Since the data is already very clean, the only preprocessing steps tested were generic normalisation and standardisation operations, show in Table 2.

None	Normalisation	Standardisation
75.46%	74.37%	75.46%
76.27%	75.15%	76.56%
76.13%	75.03%	76.18%
76.13%	75.31%	76.07%
76.23%	75.23%	76.03%
76.04%	**75.02%**	**76.06%**

Table 2: Preprocessing comparison of RFC baselines with 5-fold CV and average accuracies

Without any conclusive increase in performance, such preprocessing was deemed unnecessary.

3 Feature Engineering

Feature correlation is known to degrade classifier performance, and by concatenating all Doc2Vec representations, information overlap is almost guaranteed.

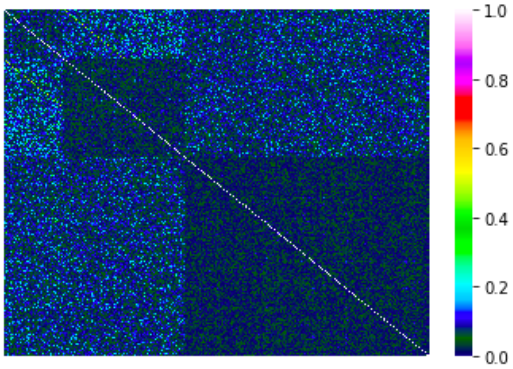

Figure 1: Correlation magnitude matrix before feature selection

[1] (Mukherjee et al., 2013) and (Rayana and Akoglu, 2015)

[2] (Le and Mikolov, 2014)

This can clearly be seen by plotting a feature correlation heatmap (Figure 1), where three clusters are clearly visible, corresponding to the 50, 100 and 200 length Doc2Vec sets, with low correlation within each group and higher correlation between groups. The data should therefore be regularised, so LASSO was used to force attribute drop out.

The coefficient given to the L1 norm sum, C, was varied, and cross validation was performed with average accuracies taken (Figure 2). A convex relationship emerges from the model performing poorly when aggressive feature selection loses too much information and when features are inadequately pruned; $C = 0.006$ results in the maximal 78.34% accuracy.

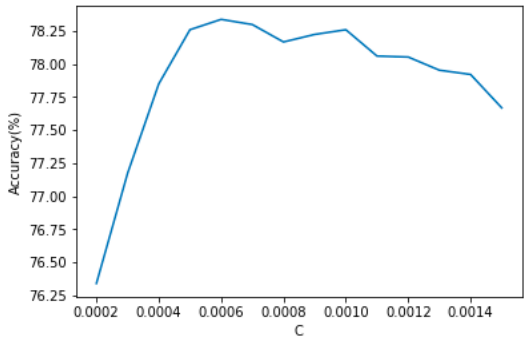

Figure 2: Accuracy vs regularisation weighting parameter

Lastly, the meta attributes were tested for relevance but found to be the least relevant of all attributes and so were discarded.

Decent attempt at choosing which features to use/encode

4 Model Selection

As an initial exploration, various standard classifiers were tested with 5 fold cross validation (see Table 3). This is not an entirely fair comparison the suitability of each model's default parameters might vary, but it will at least allow for short listing for further testing. As star ratings are inherently numeric, it was hypothesised that regression might be able to better capture the relationship between the output classes, so a custom regression-based classifier was also tested, rounding the response variable to perform the final classification. Surprisingly, this model performed very poorly. This suggests that the underlying emotional relationships could actually supersede the simplistic numeric quality: the shift from one star to three might actually be fundamentally very different from the shift from three to five. One example is that one star and five star reviews are both very polarised and so might observe higher incidences of intensifiers and stronger language, whereas three star reviews might be generally more neutral, and so for this aspect of language the former classes might actually be closer than to three stars.

Shouldn't we try it anyways at the end?

It's nominal, and it's different from numeric where 3 stars + 2 stars ≠ 5 stars

I feel like this is the wrong justification, it's not really "emotional"...

Basically, nominal data

Classifier	Average accuracy
AdaBoost	78.18%
Multilayer Perception	77.84%
K-NN	77.65%
SVM	82.81%
Decision Tree	66.48%
LDA	82.09%
Gaussian NB	73.75%
Gradient Boosting	80.72%
Rounded RF Regressor	71.21%

Table 3: Average performance of various untuned classifiers on feature selected data

Could've done well to hypothesise why, based on theories in class – svm best on sparse data and can alter kernel to fit non-linear

By a significant margin, the support vector machine, linear discriminant analysis and gradient boosting classifiers performed best, with a 2.5 - 4.5% increase in accuracy over the random forest classifier, so these were selected to be tuned.

5 Hyperparameter Tuning

5.1 Linear Discriminant Analysis

While performing very commendably out of the box, as an algorithm geared largely towards dimensionality reduction, linear discriminant analysis is the least parameterised of the three selected models, with essentially only the choice of solver and degree of shrinkage causing any significant difference in performance. All sane combinations were tested and plotted against five fold average accuracies in Figure 3.

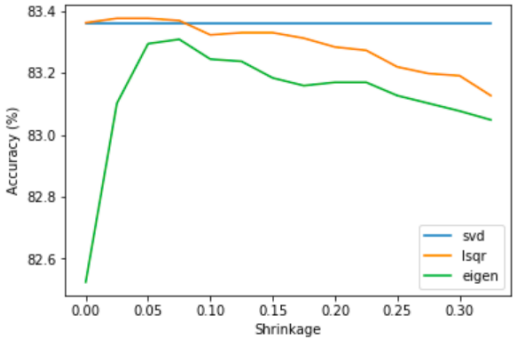

Figure 3: LDA solver and shrinkage 5CV average accuracies on feature selected data

The optimal choice of parameters was thus found to be the least squares solver with 0.05 shrinkage; when trained on all training data this yielded a final test score of 83.99%.

Decent attempt at hyperparameter tuning

5.2 Gradient Boosting Classifier

Out of the three chosen models, tuning the gradient boosting classifier is the most involved due to the high number of parameters: navely searching over such a high dimensioned space is extremely computationally costly. One solution is

hypothesises the behaviour of the hyperparameters and its correlation with the other hyperparameters

tuning subsets of the parameter space sequentially. In doing so however, tuning order is important as parameters may be highly dependent; there are therefore three principles to adhere to: parameters tuned separately should be as independent as possible, the most critical parameters should be tuned first, and parameters not yet tuned should be initially assigned reasonable guesses. A proven scheme is {learning_rate, n_estimators} as the first set, then three sets for the tree affective parameters {max_depth}, {min_samples_split, min_samples_leaf} and {max_features}, finally tuning boosting affective {subsample}.[3]

Great strategy

A learning rate often has a corresponding optimal number of estimators, forming an inverse relationship, so number of trees was varied on relatively high learning rates of 0.1 and 0.2. Unfortunately, the upper endpoint performed best even when testing up to four hundred trees; in principle it would be best to keep testing higher numbers but due to limitations of computational power a limit of one hundred trees was imposed during further tuning.

Tuning max depth over the range of five to seventeen maxed out at seventeen; this suggests that the true optimal max depth is greater than seventeen but as the increase in accuracy between fifteen and seventeen was negligible seventeen was chosen. This increased the accuracy by almost 2% to 82.71%.

Very good observations for each hyperparameter tuning, each properly balanced with practicality, effectiveness and efficiency. Great observations but not very in-depth. Perhaps too much emphasis on what has been observed, rather than why this has been observed and why it works

Tuning minimum samples per leaf and per split over the ranges 1000 - 2000 and thirty to seventy respectively resulted in optimal values of 1800 and sixty respectively, well within their ranges. This increased the accuracy by 0.3% to 83.01%.

Tuning max features around the square root of the number of features, ten to thirty, performed best at twenty five. This did however decrease the accuracy to 82.94%, likely an effect of evaluator variance, suggesting that the gains were minimal.

Lastly tuning the subsample value between 0.6 and 0.95, the optimal value was the already used 0.8, thus accuracy remained unchanged.

The model was finally trained on all test data with 100 - 20k estimators and learning rates based on oob improvement. 400 trees performed best, suggesting that higher estimator counts may have been overfitting. The final accuracy was 83.42%.

5.3 Support Vector Machine

Tf you doing bro....

An SVM is similar to LDA in that there are comparatively fewer parameters to tune, but unfortunately SVMs tend to take significantly longer to train. In particular, one single fold of a linear kernel model with soft margin parameter $C = 10$ required over five hours. This renders brute searches infeasible, so instead a variety of parameter combinations were run over the course of a couple days to be inspected for tighter tuning using binary search (see Figure 4).

The sigmoid kernel seemed limited to about 82.5% accuracy, and was thus rejected. The positive gradient of the polynomial kernel models demonstrate potential for greater performance, however models with the necessarily high C would infeasibly take weeks to train. The remaining two kernel functions, linear and the radial basis function, both performed very similarly, but the convex shape of the rbf models' accuracies is suggestive of a higher extremum; hence, only rbf kernels were further searched with C in the vicinity of one. The optimum was found to be at $C = 0.8$, yielding a final test accuracy of 84.47%.

[3](Jain, 2016)

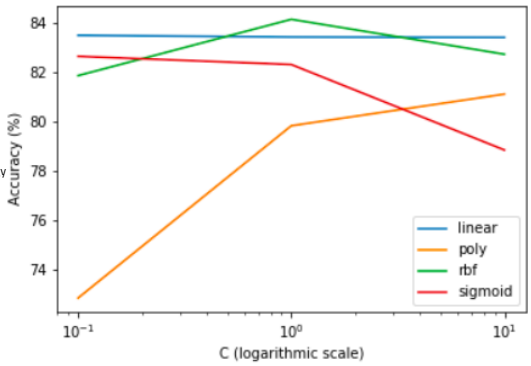

Figure 4: Average accuracies of SVM kernels and soft margin coefficients on feature selected data

Very good choices based on the results you have gotten, but I am honestly shocked to hear the 5 hours part.... like how

6 Conclusion

Three models were ultimately chosen, tuned and fitted to the reviews, predicting star ratings with accuracies of 83.99% using linear discriminant analysis, 83.42% using a gradient boosting classifier, and the highest performing 84.47% using a support vector machine model. Due to computational limitations, tuning was not able to be performed to the fullest extent on these models, so one major improvement would be to further improve parameters through more thorough cross validation searches; combining these models using stacking into one ensemble classifier could also increase performance.

References

Nothing on error analysis, like evaluation bias and model bias

Aarshay Jain. 2016. Complete machine learning guide to parameter tuning in gradient boosting (gbm) in python. `https://www.analyticsvidhya.com/blog/2016/02/`.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *arXiv:1405.4053 [cs.CL]*.

Arjun Mukherjee, Vivek Venkataraman, Bing Liu, and Natalie Glance. 2013. What Yelp fake review filter might be doing? In *7th International AAAI Conference on Weblogs and Social Media*.

Shebuti Rayana and Leman Akoglu. 2015. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 985–994.