Predicting Customer Satisfaction from Online Product Reviews

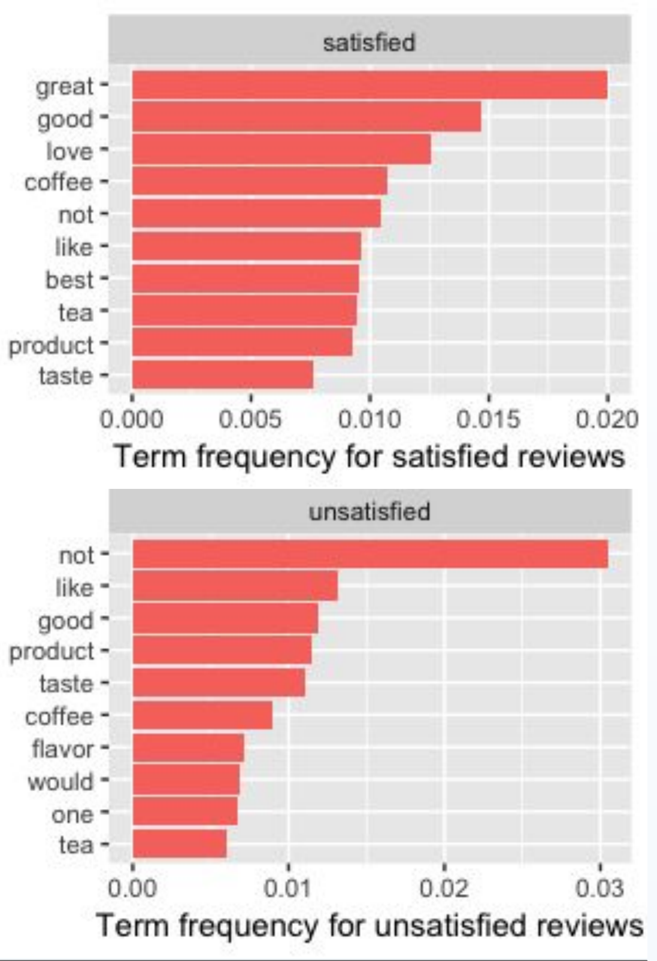
Shawn Ban & Julia Brown

Task Description and Motivation

- Sentiment analysis is a highly influential and expanding area of Natural Language Processing, with wide ranging applications. E.g. Predicting customer satisfaction from unlabeled data could help inform business decisions by better understanding how a product is received
- We optimise a machine learning pipeline to predict the sentiment of online product reviews as either satisfied or unsatisfied, evaluate our model using key metrics Accuracy, AUC and processing time and compare our findings with previous research in this area [6]

Dataset Description and Initial Exploration

- The dataset, from City University Data Repository, comprises 568,454 online product reviews
- There are 10 features; 7 non-predictive (*ID, Product ID, User ID, Profile Name, Time, Helpfulness Numerator, Helpfulness Denominator*), 2 predictive (*Summary, Text*) and 1 Dependent (*Score*). We drop the non-predictive features and merge the 2 predictive
- We remove 2,516 reviews with missing or nonsensical scores, leaving 565,938 instances
- The dependent variable *Score* is nominal, taking values 1 through 5, with 1 being Very Poor and 5 being Excellent. To produce a binary classification task, we label all reviews with scores 4 or 5 as *satisfied* and all reviews with scores 1,2 or 3 as *unsatisfied*
- The distribution of classes is somewhat imbalanced, with 440,759 satisfied/positive reviews (77.9%) and 125,179 unsatisfied/negative reviews (22.1%)



Hypothesis

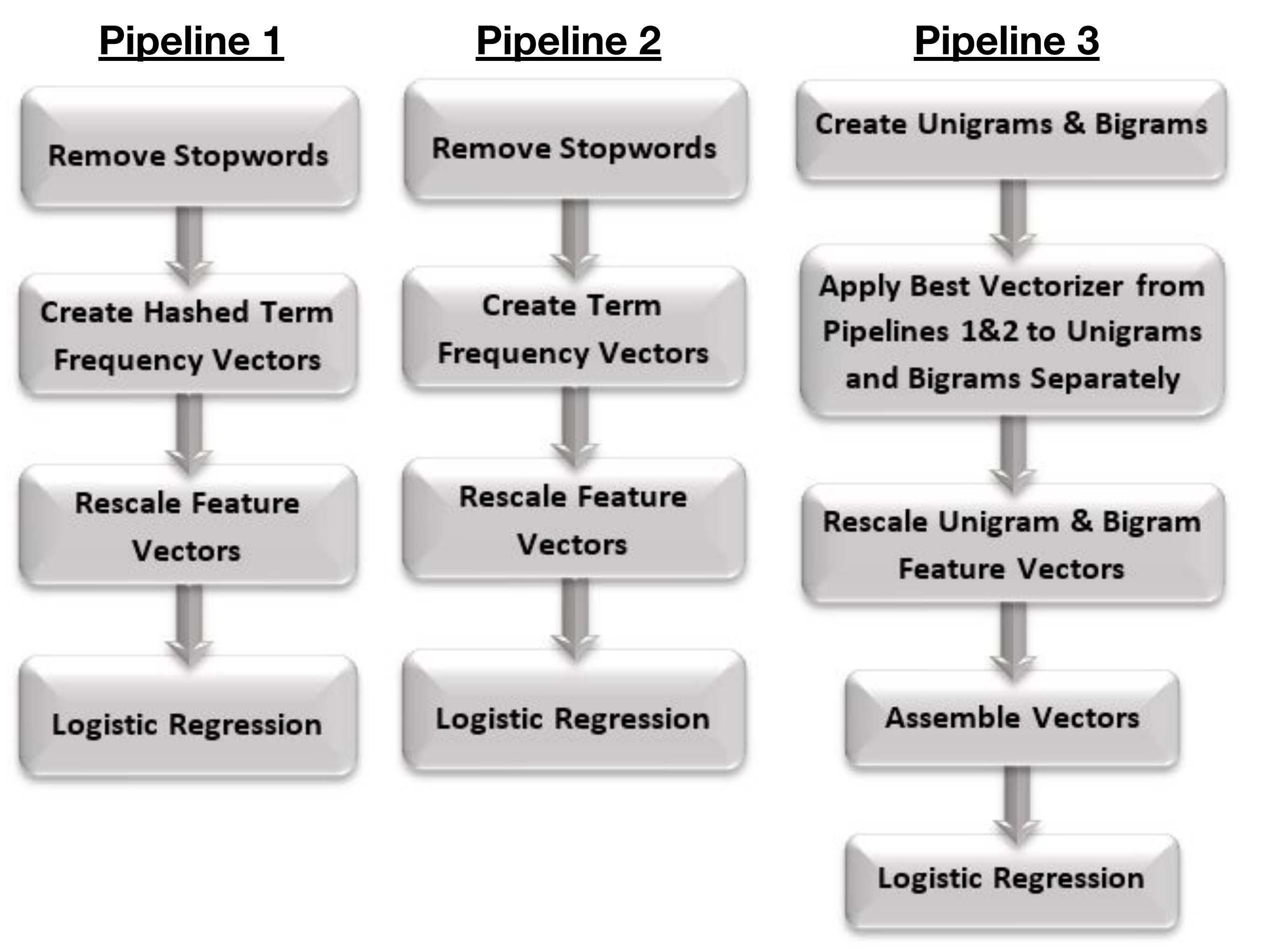
As with previous research into machine learning for sentiment analysis, we expect a high level of performance that can be moderately improved through parameter optimisation and experimentation [6]. We anticipate that representing the text as both unigrams and bigrams will improve accuracy [4], though the performance gain may be insufficient to compensate for the additional computation time. It is also expected that a larger hashing vector size and larger training set will yield greater accuracy due to reducing the number of collisions [5] and reducing the influence of noise in the data respectively.

Methodology and Parameter/Experiment Choices

- We held out 20% of the dataset for testing and split the remainder into various training subsets for experimentation. We use the largest subset (70%) for the first two experiments and all the subsets for the third experiment (see below)
- Logistic regression was selected as the most suitable classification algorithm [2] and 3 different pipelines were constructed to observe the effect of different pipeline stages
- The data fed into the pipelines was first tokenized, with capitals and punctuation removed
- We utilise paramGrid to tune the regularization parameter and assess the impact of bias/variance trade-off, with λ = 0.3;0.1; 0.01. The best parameter value was determined using 20% of the training set for validation
- We hold Elastic Net regularization constant throughout as initial experimentation revealed α = 0 (Ridge penalty) gave consistently better results
- We perform 3 experiments to determine the effect of different processing choices and training set sizes:

Experiment 1 (pipelines 1 and 2):
Comparison of HashingTF vs Count Vectorizer (CV)
Effect of vector size using N = 1024, 4096, 8192
Experiment 2 (pipelines 2 and 3):
Comparison of unigrams without stopwords vs unigrams + bigrams with stopwords
Experiment 3 (pipeline 3):
Effect of training set size using 70% (396,137), 7% (39,434), 2% (11,376), 0.9% (5,150), 0.1% (537)

- We assess performance on the testing set (held constant throughout) using three metrics: Accuracy (% of reviews classified correctly), AUC (area under the receiver operating characteristic curve) and time taken (wall time to train 3 models). Precedence was given to AUC, which offers a more robust measure when there is a large class imbalance


Pipeline 1
Remove Stopwords
Create Hashed Term Frequency Vectors
Rescale Feature Vectors
Logistic Regression
Pipeline 2
Remove Stopwords
Create Term Frequency Vectors
Rescale Feature Vectors
Logistic Regression
Pipeline 3
Create Unigrams & Bigrams
Apply Best Vectorizer from Pipelines 1&2 to Unigrams and Bigrams Separately
Rescale Unigram & Bigram Feature Vectors
Assemble Vectors
Logistic Regression

Experimental Results

Experiment 1: HashingTF vs. Count Vectorizer (CV) for different N

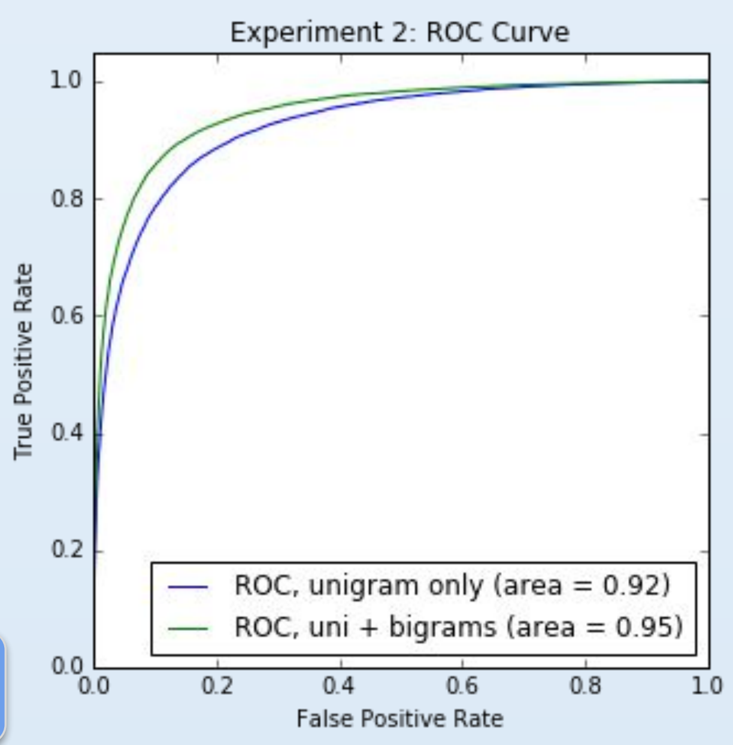
	N = 1024		N = 4096		N = 8192	
	Hashing	CV	Hashing	CV	Hashing	CV
Accuracy	83.7%	86.1%	86.5%	87.9%	87.4%	**88.5%**
AUC	86.1%	90.0%	90.1%	91.9%	91.1%	**92.3%**
Regularization	0.01	0.01	0.01	0.01	0.01	0.01
Wall Time	34min 15s	42min 52s	**33min 12s**	42min 19s	33min 48s	42min 23s

- CV consistently outperforms hashing: hashing creates collisions while CV drops infrequent words, this implies that infrequent words have low predictive power so dropping them is preferable to introducing collisions
- Hashing performance improves markedly as vector length increases, due to less frequent collisions
- Vector length also improved performance for CV
- CV was more computationally expensive and requires finding a cut-off point to drop low frequency words
- We use N = 8192 and CV for subsequent experiments

Experiment 2: Adding Bigrams

	N = 8192, CV	
	Unigrams only	Uni + Bigrams
Accuracy	88.5%	**90.6%**
AUC	92.3%	**94.7%**
Regularization	0.01	0.01
Wall Time	**42min 23s**	1h 17min

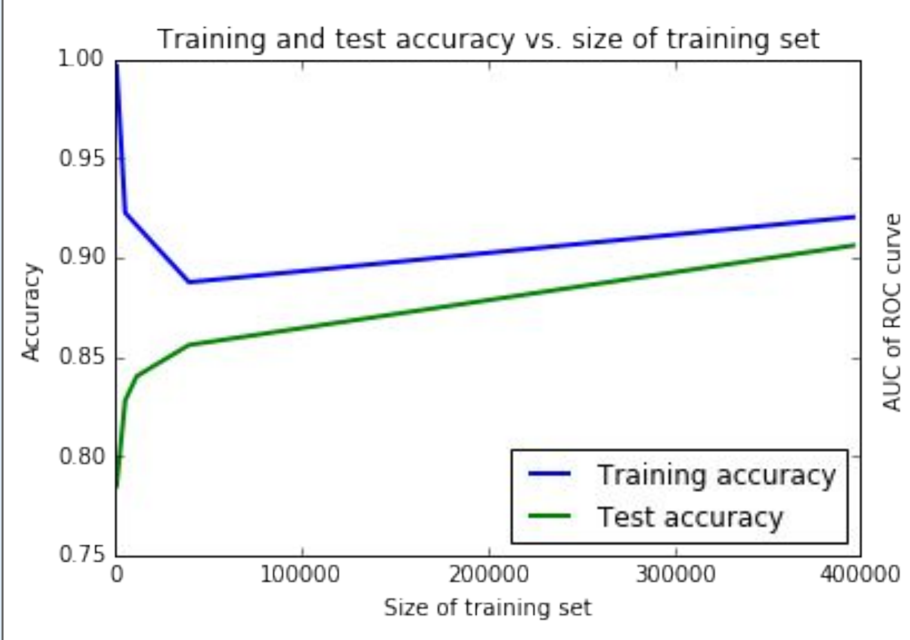
- Stopwords were not removed as this creates false bigrams
- Adding bigrams increased performance by 2.4 percentage points, but took almost twice as long computationally
- Depending on dataset size, this may be a feasible feature extraction step to boost performance

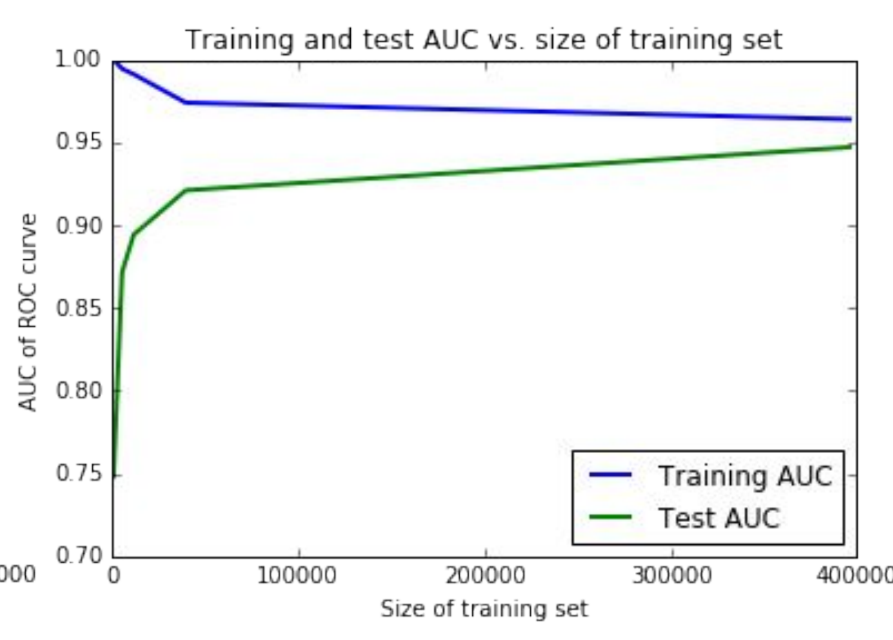


Experiment 3: Effect of Training Set Size *n*

No. of Instances	**396,137**	**39,434**	**11,376**	**5,150**	**537**
Training Accuracy	92.0%	88.8%	91.6%	92.3%	**99.6%**
Test Accuracy	**90.6%**	85.6%	84.0%	82.8%	78.5%
Training AUC	96.4%	97.4%	99.1%	99.4%	**100.0%**
Test AUC	**94.7%**	92.1%	89.4%	87.1%	74.8%
Regularization	0.01	0.3	0.3	0.3	0.01
Wall Time	1h 15min	**1h12min**	1h15min	1h 16min	1h 15min

- Test accuracy and AUC improved logarithmically with training set size
- The large divergence between training and test accuracy and AUC indicates severe model overfitting using small datasets
- Overfitting particularly pronounced below *n*=11,376 (2% of data)
- For large *n*, there are diminishing marginal returns to adding more data; further feature extraction should be used instead to improve performance
- Training time does not increase significantly with training set size, however choice of feature extraction does have an impact, see experiment 2





Best Parameters and Set-Up:		
Training Instances: 396,137	Count Vectorizer	Unigrams
Regularization Parameter: 0.01	Vector Size: 8,912	+ Bigrams

Critical Analysis and Evaluation

- Model performance exceeded expectations, producing an Accuracy of 90.6% and AUC 94.7% - greater than that achieved by Pang et al. when using Naive Bayes and SVM for sentiment classification [6]
- Varying the size of the training data shows the logarithmic improvement in performance with training set size, suggesting that performance can be improved by using substantially larger training corpus; mirrors findings by Banko and Brill [1]. A larger corpus also help ensure model generalisability by reducing overfitting
- Correctly classifying 90% of reviews should meet business needs and far exceeds performance when using a lexicon of buzzwords devised by humans [6]
- Experimentation and parameter tuning proved valuable, delivering an improvement of 12 percentage points on the lowest Accuracy and 20 percentage points on the lowest AUC.
- Acceptable error rate and computation time should be considered when choosing an optimal model: a trade-off must often be made between improved performance and increased computation time, evident here when comparing the use of unigrams alone with unigrams + bigrams
- The class imbalance did not adversely affect performance, possibly due to prudent feature extraction: e.g. although stopwords were removed in many experiments, '*not*' was retained as the word distributions between classes revealed this would be a differentiating feature. This also highlights the importance of initial dataset exploration
- Binary classification may oversimplify the task; reviews with a score of 3 may not be unsatisfied and introducing a neutral class has been shown to improve results [3]
- Document level sentiment analysis may oversimplify; often customers give mixed feedback regarding different aspects of a product; de-constructing reviews to analyse different issues would be more use commercially
- Lack of model transparency prevents key indicators of dissatisfaction from being identified

Conclusions and Future Work

- We developed a model able to predict customer satisfaction 90% of the time
- We discovered feature extraction e.g. adding bigrams, choice of input representation e.g. CV and training set size can all significantly improve performance
- For future work, additional feature extraction and pre-processing could be applied, such as sentiment score based on a customized lexicon, e.g. train on adjectives only, neutralise spelling and apply word stemming
- We could further compare our model with other state-of-the-art text classification methods (decision trees, neural networks, support vector machines) and features (odds ratio, Gini index, information gain) [4]
- A regression analysis could be conducted to determine how far wrong the incorrectly classified reviews were (e.g. predicting 1 vs. ground-truth of 5) by calculating mean-squared error of predictions

References

[1] Banko, M. and Brill E. (2001). Scaling to Very Very Large Corpora for Natural Language Disambiguation, *Proceedings of 39th Annual Meeting of the Association for Computational Linguistics*, pp. 26-33
[2] Hamdan, H., Bellot, P., and Bechet, F. (2015). Sentiment lexicon-based features for sentiment analysis in short text, *Research in Computing Science,* 90 pp.217-226
[3] Koppel, M. and Schler, J. (2006). The importance of neutral examples for learning sentiment, *Computational Intelligence*, 22, pp. 100-109
[4] Mironczuk, M.M., Protasiewicz, J. (2018). A recent overview of the state-of-the-art elements of text classification, *Expert Systems With Applications*, 106 pp. 36-54
[5] Mitzenmacher, M. and Vadhan, S. (2008). Why simple hash functions work: exploiting the entropy in a data stream, *Proceedings of the 19th annual ACM-SIAM symposium on Discrete algorithms*, pp.746-755
[6] Pang, B., Lee, L. and Vaithyanathan, S. (2002). Thumbs up? Sentiment Classification using Machine Learning Techniques, *Proc. of the 7th ACL-02 Conference on Empirical Methods in Natural Language Processing* pp. 79-86