Sentiment Classification

TAE (Fall 2019), Team 38: Seah Ee Song, Ku Wee Tiong

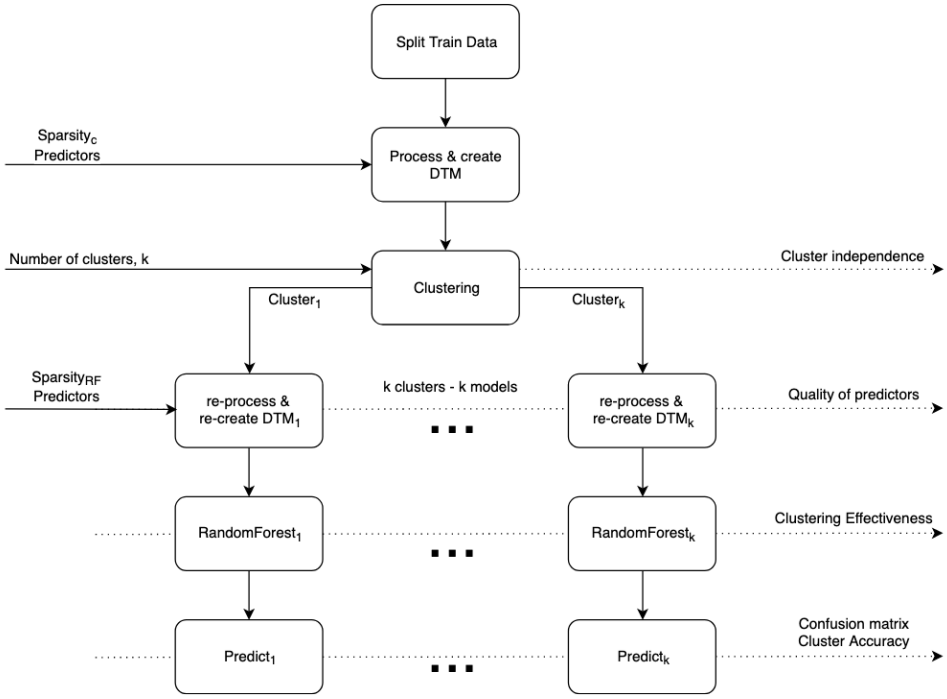
Introduction

The text data provided in the form of a tweet contains between 0 and 140 characters. Embedded within this short prose is the tweeter's sentiment – how he/she is feeling and reacting at the moment of writing. We train a machine to be classify the writer's sentiment between three categories: Negative, Neutral, Positive.

Sentiment classification is the task of looking at a piece of text and telling if someone likes or dislikes the thing they're talking about.

Methodology

The high-level schematic of our model is provided in Fig 1 and elaborated below.



Processing

When training a model, we first split the data into train and validation sets with ratio 0.8. We then process the data by standardizing the alphabetical case and removing elements in the tweet that may not be helpful in discerning the tweeter's sentiment with the text mining (*tm*) library in R. These include stop words, punctuation and numbers. However, we found that the default stopwords of the tm library contain stopwords like "aren't" and "not" which may reveal the tweeter's sentiment,. Hence, we modified the stopwords so that these words would be included in the tweets. Finally, we stem the documents into their root words.

The *RWeka* library allows us to tokenize documents into N-grams. Making use of this, we create our document term matrix (dtm) with both single words and two-word phrases. Finally, we reduce the sparsity (sparsity_C) of our matrix to capture only words and phrases (predictors) that are frequently found.

Clustering

Studying the tweets, we saw some obvious patterns in the data that help to segregate them into clear clusters such as consistent weather reports. By clustering and training models on each cluster, we hope to improve predictive power. We chose k=3 after repeated iterations, balancing between achieving helpful segregations and overfitting the data.

Random Forest

After clustering, we mark each row by its cluster and regenerate a new dtm for each cluster, with different settings (especially sparsity_RF). Hence, three new dtms are created, which will be used for a Random Forest model each.

Model Selection

Before fixing on Random Forest, we had tried other simple models like CART and Bayesian. Repeated trials suggested that Random Forests produce the best performance for the classification of the dtm created.

Results

Model Refinement

To refine our model meta-parameters, we trained multiple models and tested them on the validation (val) set produced. Using the same seed for the models, we were able to compare the accuracies produced and determine the relative benefit a change in some parameter brings.

We only made modifications to a small subset of parameters which we deem key for our model. They are listed as the column headers of the following table. The resulting accuracies when testing on the val set are given on the right side of the table.

Model numbers are not necessarily in chronological order as multiple models were trained concurrently. For example, we began with model 1 and 2 and decided that model 1 was better Then we produced 1.1 and 1.2, finally deciding that 1.1 was better. Note that accuracies for clusters are not comparable across rows due to different sparsity values which result in different row data and hence different clusters forming. However, overall accuracies are comparable.

Model	Stopwords	Sparsity for C Stage	Sparsity for RF Stage	ntree	Acc. C1	Acc. C2	Acc. C3	Overall Acc.	Direction
2	'english'	.98	0.998	200	100	82.36	81.67	83.13	
1	Subset of 'english'	.98	0.998	200	98.8764	100	82.97	84.75	Use subset of 'english'
1.1	Subset of 'english'	.98	0.999	200	100	83.03	99.10	84.95	Higher sparsity for RF is better
1.2	Subset of 'english'	.97	0.998	200	100	82.93	98.24	84.84	Lower sparsity for C is better

1.2.1	Subset of 'english'	.95	0.998	200	82.59	100	84.74	84.24	Do not lower sparsity for C beyond .97
1.1.1	Subset of 'english'	.95	0.999	200	82.15	100	85.43	84.08	Do not lower sparsity for C beyond .97
1.1.3	Subset of 'english'	.97	0.999	500	100	83.25	98.24	**85.12**	Use this model for submission

Table 1: Table of models of different meta-parameters attempted and the resulting accuracies. This table captures only a subset of refinement activity.

Findings

With reference to Fig 1, we found that having a lower sparsity_C in the Clustering stage improves accuracy by 0.1 percent, a modest improvement. This may be because having fewer terms reduces noise, enabling cleaner clusters to form. However, setting sparsity_C too low removes that benefit. There is typically one cluster with 100% accuracy. This cluster is made up of weather reports.

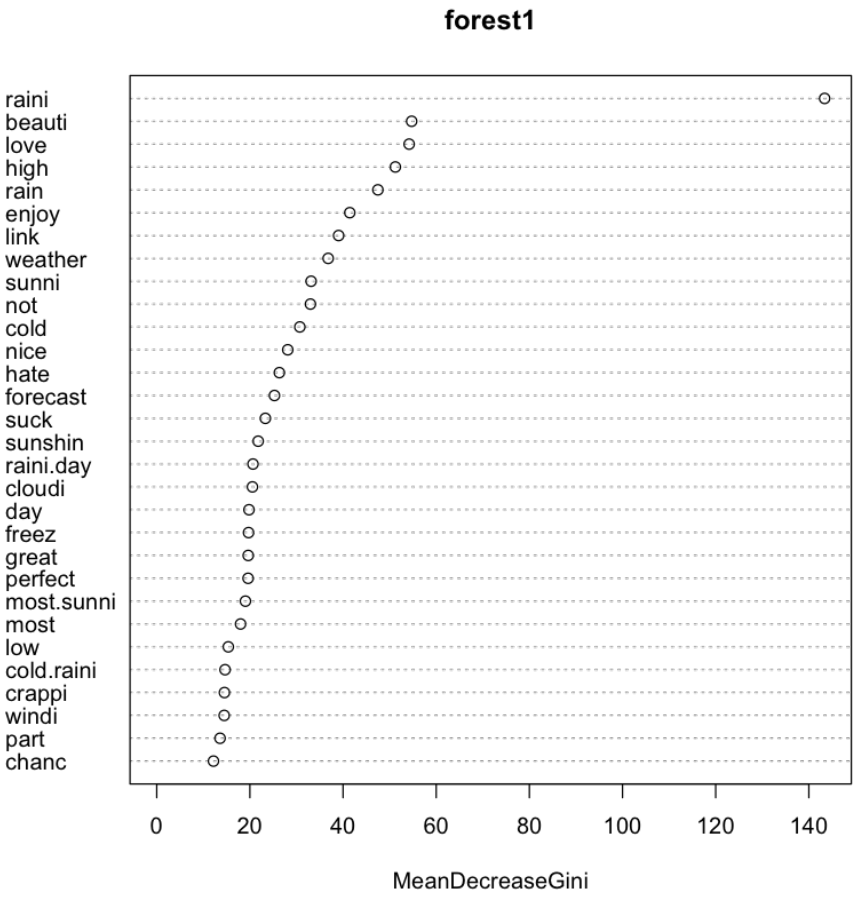
We also found that having higher sparsity_RF in the Random Forest stage increases accuracy by 0.2 percent. This may be because more predictors are fed to the Random Forest model, increasing its predictive ability.

The most important finding is that stopwords modification result in a relatively large 1.6% increase in accuracy. The full list of stopwords removed from the default set is found in the source code. Even though more improvements could possibly be made, time constraints dictate that we end our iterations at model 1.1.3.

Interpretability and Limitations

The model is interpretable in several ways.

- Clusters of tweets are formed. One cluster that consistently forms is that of weather reports due to their close similarity. They have neutral sentiment. Other clusters may be formed around certain weather patterns like 'storms' and 'rain' versus 'sunshine' and 'hot'. Although these clusters are clear to the human eye, sentiment is still challenging to gauge. Sunshine is not always preferred and rain is not always detested.
- Variable importance in Random Forests. While the exact hierarchy of trees in the forest may be difficult to visualise, one can still tell the relative importance of words by the mean decrease in Gini metric. Fig 2 shows an example of a Random Forest produced during training on a cluster. We see that 'raini', coming from 'raining', holds a distinctively high mean decrease in Gini. We also see that some stopwords we re-included in training hold some importance, such as 'not'.



The team would like to suggest the following for future work:

- Emoticons as predictors. Emoticons present a clearer indication of a tweeter's sentiment. A computer may have difficulty discerning sarcasm from plain words, but this would be revealed through emoticons. However, emoticons come as permutations of punctuation and may be challenging to keep as a word while removing punctuation elsewhere.
- Hashtag words as predictors. Similar to emoticons, hashtags can provide clearer indications of a tweeter's sentiment which may contradict what plain words indicate.