# COMP30027 Assignment 2: Report

**Anonymous**

## 1 Introduction

Sentiment analysis focuses on classifying text into sentiments. My goal is to apply sentiment analysis to develop and train three reliable sentiment classifiers and one baseline for Twitter posts (tweets). This involves extracting and selecting useful features from a dataset of tweets, then choosing and evaluating highly reliable classifiers.

### 1.1 Dataset

The dataset provided contains two lists of tweets/instances made prior to 2017 (Rosenthal et al., 2017). The training set in `Train.csv` contains 21802 labelled instances and the testing set in `Test.csv` contains 6099 unlabelled instances. For each instance, included is the text and tweet ID. The tweets included vary in content. For example, some are not in English: "*season in the sun versi nirvana rancak gak..slow rockkk...*". Tweets can either be "positive", "neutral" or "negative", distributed as shown in Figure 1.

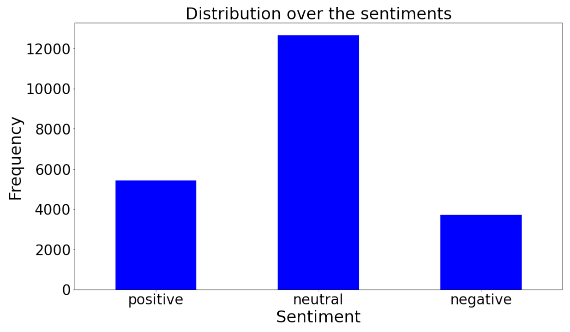

Figure 1: Training sentiment distribution

## 2 Methodology

The following methods are developed with reference to prior works on Twitter sentiment analysis (Go et al., 2009; Bird et al., 2009; Barbosa and Feng, 2010).

### 2.1 Instance cleaning

Some features extracted rely on the text in the tweets being cleaned. Cleaning involves removing stopwords (Section 2.1.1), links, hashtags, mentions, numbers, non-alphanumeric characters. Also performed is the reduction of repeated letters with more than two occurrences, as suggested by Go et al. (2009).

#### 2.1.1 Stopwords

Manual stopword list construction is tedious and inexhaustive (consider other languages). Instead, I start with the Python Natural Language Toolkit's (`NLTK`) stopword list, which includes non-English languages (Bird et al., 2009).

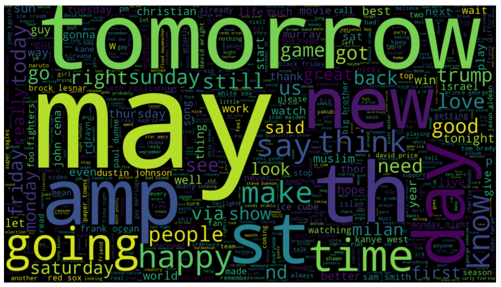

Figure 2: Word cloud with `NLTK` stopword removal

Next, based on Figure 2, some extra words are manually added to the list, leading to the final vocabulary list in Figure 3.

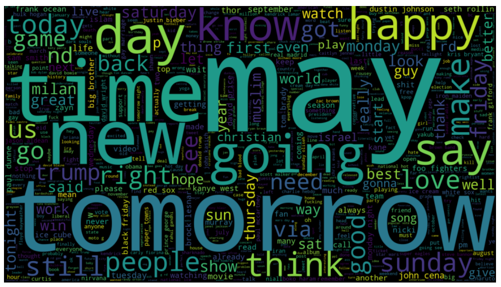

Figure 3: Word cloud with modified `NLTK` stopword removal

### 2.2 Feature extraction

These features are all chosen for their potential correlations with sentiments.

#### 2.2.1 Twitter features

First, I extract platform-specific features from the tweets: hashtags, mentions, and links (Go et al., 2009; Barbosa and Feng, 2010). These are integrated within the platform, meaning they are widely used (commonplace in tweets).

#### 2.2.2 Linguistic features

Next, linguistic features are extracted and used to tokenize the tweet in different ways: part-of-speech tags, lemmas, lemma 2-grams, phonetics (phenomes of words), punctuation, and emoticons (Barbosa and Feng, 2010; Bird et al., 2009; Weide, 1998). Lemmas are preferred over words, as they group words of the same root (Bird et al., 2009).

Emoticons are identified as string combinations of eye, middle and mouth characters, using a different method to Go et al. (2009), since they miss many emoticons, such as the crying `:'(`. Emoticons are then categorized into happy `:)`, sad `:(`, neutral `:|` or surprised `:o`.

#### 2.2.3 Metric features

These are numeric counts of feature types within a tweet: words, characters, alphabetic characters, links, hashtags, mentions, and emoticons. Also considered are average word length and if a tweet is quoting another.

#### 2.2.4 How many features per feature type is enough?

With over 20,000 unique instances in the training dataset, many feature types generate a large number of unique features (e.g. a large vocabulary of lemmas). Minimizing the feature space with a limit of features per feature type reduces model complexity. To determine this number $M_f$, the top candidate classifiers are compared for each of $M_f \in \{10, 100, 1000, 5000\}$ by the accuracies found using Section 2.4.1.

#### 2.2.5 Vectorization

At the time that the data was collected, tweets were limited to 140 characters (Pardes, 2017). This means that features appear once (at most twice) in a tweet. Therefore, non-metric features are vectorized by occurrence counts, as TF-IDF is not applicable with near-binary features.

### 2.3 Classifier Selection

#### 2.3.1 Baseline

The baseline model used is the 0-R: most frequent class (Classifier 4). The three chosen models (Classifiers 1, 2, and 3) need to outperform this.

#### 2.3.2 Classifier Candidates

The following classifier/parameter combinations are considered for the final three models.

**Multinomial/Bernoulli Naive Bayes**:

| Parameter | Options |
|---|---|
| Include Priors | Yes, No |
| $\alpha$ Smoothing | 0, 1, 10 |

Table 1: Naive Bayes Parameters

**Logistic Regressions**:

| Parameter | Options |
|---|---|
| Fit Intercept | Yes, No |
| Maximum Iterations | 100, 500 |
| Optimization | `sag`, `saga` |

Table 2: Logistic Regression Parameters

**Decision Trees**:

| Parameter | Options |
|---|---|
| Maximum Depth | 1, 100, 500 |

Table 3: Decision Tree Parameters

$K$ **Nearest Neighbours**:

| Parameter | Options |
|---|---|
| $K$ | 1, 10, 100, 500 |
| Vote Weighting | `uniform`, `distance` |

Table 4: K-Nearest Neighbour Parameters

**Support Vector Classifiers**:

| Parameter | Options |
|---|---|
| Kernel | Linear, Cubic |
| Regularization $C$ | 0.1, 1, 3 |
| Decision Function | One-v-One, One-v-Rest |

Table 5: Support Vector Classifier Parameters

## 2.4 Evaluation

### 2.4.1 Which Parameter Configurations Are Best?

Of the combinations in Section 2.3.2, the top three (Classifiers 1, 2, and 3) are determined with the following scoring system. First, cross-validation scores are calculated on the training set. Next, cross-validation is performed on the maximum subset of data with a uniform sentiment distribution. The 10 scores are averaged $\bar{x}$, and their standard deviation $s$ found. The best candidate configurations have the highest bound for the right-sided 95% confidence interval of mean cross-validation scores:

$$\begin{aligned} 95\% \text{ C.I. Bound} &= \bar{x} - \mathbb{F}_{t_9}^{-1}(0.95) \times \frac{s}{\sqrt{n}} \\ &= \bar{x} - 1.8331 \times \frac{s}{\sqrt{10}} \end{aligned}$$

where $t_9$ is the $t$-distribution with 9 degrees of freedom and $n = 10$.

### 2.4.2 Evaluation Metrics

Accuracies, precisions, recalls and $F_1$ scores (per sentiment) are generated for each classifier that makes it past Section 2.4.1.

### 2.4.3 Which models are label distribution agnostic?

It is unknown whether the `Test.csv` and `Train.csv` sentiments are similarly distributed, so evaluation metrics from Section 2.4.2 are calculated on four 4 : 1 train/test splits.

Where "Given" denotes a set of data with sentiment distributions similar to those in Figure 1, and "Uniform" denotes a maximum subset of data with uniformally distributed sentiments, evaluation metrics are generated on the train/test combinations in Table 6. The metrics are then compared to determine which models are agnostic to sentiment distributions.

| Training | Testing |
|---|---|
| Uniform | Uniform |
| Given | Given |
| Given | Uniform |
| Uniform | Given |

Table 6: Training and testing splits

## 3 Results

### 3.1 Highest cross-validation accuracy per number of maximum features

After generating classifiers for each of the $M_f$ values, the following boasted the highest scores calculated in Section 2.4.1.

| Classifier | Logistic Regression |
|---|---|
| Parameters | `sag` Optimizer, Without Intercepts & 500 Max Iterations |
| $\bar{x}$ | 0.5491 |
| $s$ | 0.05083 |
| 95% C.I. Bound | 0.5196 |

Table 7: $M_f = 10$ Top classifier

| Classifier | Logistic Regression |
|---|---|
| Parameters | `sag` Optimizer, With Intercepts & 500 Max Iterations |
| $\bar{x}$ | 0.5852 |
| $s$ | 0.03187 |
| 95% C.I. Bound | 0.5667 |

Table 8: $M_f = 100$ Top classifier

| Classifier | Bernoulli N.B. |
|---|---|
| Parameters | With Priors, $\alpha = 5$ |
| $\bar{x}$ | 0.6179 |
| $s$ | 0.01537 |
| 95% C.I. Bound | 0.6090 |

Table 9: $M_f = 1000$ Top classifier

| Classifier | Bernoulli N.B. |
|---|---|
| Parameters | With Priors, $\alpha = 1$ |
| $\bar{x}$ | 0.6372 |
| $s$ | 0.01438 |
| 95% C.I. Bound | 0.6288 |

Table 10: $M_f = 5000$ Top classifier

| Classifier | Bernoulli N.B. |
|---|---|
| Parameters | With Priors, $\alpha = 1$ |
| $\bar{x}$ | 0.6361 |
| $s$ | 0.01028 |
| 95% C.I. Bound | 0.6301 |

Table 11: $M_f = 10000$ Top classifier

It appears that *accuracy* $\propto M_f$. The final classifer models are trained with $M_f = 10000$ to maximize accuracy.

### 3.2 Top 3 parameter/classifier configurations for $M_f = 10000$

The following classifiers have the highest scores calculated in Section 2.4.1:

| Classifier Parameters | Bernoulli N.B. With Priors, $\alpha = 1$ |
|---|---|
| $\bar{x}$ | 0.6361 |
| $s$ | 0.01028 |
| 95% C.I. Bound | 0.6301 |

Table 12: Classifier 1

| Classifier Parameters | Bernoulli N.B. Without Priors, $\alpha = 1$ |
|---|---|
| $\bar{x}$ | 0.6365 |
| $s$ | 0.01159 |
| 95% C.I. Bound | 0.6298 |

Table 13: Classifier 2

| Classifier Parameters | Multinomial N.B. With Priors, $\alpha = 1$ |
|---|---|
| $\bar{x}$ | 0.6343 |
| $s$ | 0.01428 |
| 95% C.I. Bound | 0.6260 |

Table 14: Classifier 3

### 3.3 Evaluation metrics for the final 4 models

The metrics describes in Section 2.4.2 are calculated for the final classifiers (Classifiers 1, 2, 3, and 4). Per-class metrics are shown in order of postive (+), neutral (n), then negative (–) sentiments.

| Classifier Parameters | Bernoulli N.B. With Priors, $\alpha = 1$ |
|---|---|
| Uniform Training | Uniform Testing |
| Accuracy | 0.62 |
| Precisions (+, n, –) | 0.65, 0.55, 0.65 |
| Recalls (+, n, –) | 0.70, 0.44, 0.74 |
| $F_1$ Scores (+, n, –) | 0.67, 0.49, 0.69 |
| Given Training | Given Testing |
| Accuracy | 0.60 |
| Precisions (+, n, –) | 0.78, 0.60, 0.50 |
| Recalls (+, n, –) | 0.14, 0.99, 0.00 |
| $F_1$ Scores (+, n, –) | 0.23, 0.74, 0.01 |
| Given Training | Uniform Testing |
| Accuracy | 0.40 |
| Precisions (+, n, –) | 0.94, 0.36, 0.90 |
| Recalls (+, n, –) | 0.19, 0.99, 0.01 |
| $F_1$ Scores (+, n, –) | 0.31, 0.52, 0.02 |
| Uniform Training | Given Testing |
| Accuracy | 0.64 |
| Precisions (+, n, –) | 0.56, 0.87, 0.48 |
| Recalls (+, n, –) | 0.82, 0.50, 0.85 |
| $F_1$ Scores (+, n, –) | 0.67, 0.64, 0.62 |

Table 15: Classifier 1 metrics

| Classifier Parameters | Bernoulli N.B. Without Priors, $\alpha = 1$ |
|---|---|
| Uniform Training | Uniform Testing |
| Accuracy | 0.62 |
| Precisions (+, n, –) | 0.65, 0.55, 0.65 |
| Recalls (+, n, –) | 0.70, 0.44, 0.74 |
| $F_1$ Scores (+, n, –) | 0.67, 0.49, 0.69 |
| Given Training | Given Testing |
| Accuracy | 0.61 |
| Precisions (+, n, –) | 0.73, 0.60, 0.60 |
| Recalls (+, n, –) | 0.16, 0.98, 0.01 |
| $F_1$ Scores (+, n, –) | 0.26, 0.74, 0.02 |
| Given Training | Uniform Testing |
| Accuracy | 0.42 |
| Precisions (+, n, –) | 0.93, 0.36, 0.95 |
| Recalls (+, n, –) | 0.23, 0.99, 0.03 |
| $F_1$ Scores (+, n, –) | 0.37, 0.53, 0.05 |
| Uniform Training | Given Testing |
| Accuracy | 0.64 |
| Precisions (+, n, –) | 0.56, 0.87, 0.48 |
| Recalls (+, n, –) | 0.82, 0.50, 0.85 |
| $F_1$ Scores (+, n, –) | 0.67, 0.64, 0.62 |

Table 16: Classifer 2 metrics

| Classifier Parameters | Multinomial N.B. With Priors, $\alpha = 1$ |
|---|---|
| Uniform Training | Uniform Testing |
| Accuracy | 0.62 |
| Precisions (+, n, −) | 0.67, 0.55, 0.62 |
| Recalls (+, n, −) | 0.67, 0.41, 0.78 |
| $F_1$ Scores (+, n, −) | 0.67, 0.47, 0.69 |
| Given Training | Given Testing |
| Accuracy | 0.61 |
| Precisions (+, n, −) | 0.79, 0.60, 0.20 |
| Recalls (+, n, −) | 0.17, 0.98, 0.00 |
| $F_1$ Scores (+, n, −) | 0.28, 0.74, 0.01 |
| Given Training | Uniform Testing |
| Accuracy | 0.42 |
| Precisions (+, n, −) | 0.94, 0.36, 1.00 |
| Recalls (+, n, −) | 0.26, 0.99, 0.02 |
| $F_1$ Scores (+, n, −) | 0.41, 0.53, 0.04 |
| Uniform Training | Given Testing |
| Accuracy | 0.63 |
| Precisions (+, n, −) | 0.58, 0.88, 0.45 |
| Recalls (+, n, −) | 0.79, 0.48, 0.89 |
| $F_1$ Scores (+, n, −) | 0.67, 0.62, 0.60 |

Table 17: Classifer 3 metrics

| Classifier | 0-R |
|---|---|
| Uniform Training | Uniform Testing |
| Accuracy | 0.33 |
| Precisions (+, n, −) | 0.00, 0.00, 0.33 |
| Recalls (+, n, −) | 0.00, 0.00, 1.00 |
| $F_1$ Scores (+, n, −) | 0.00, 0.00, 0.50 |
| Given Training | Given Testing |
| Accuracy | 0.58 |
| Precisions (+, n, −) | 0.00, 0.58, 0.00 |
| Recalls (+, n, −) | 0.00, 1.00, 0.00 |
| $F_1$ Scores (+, n, −) | 0.00, 0.73, 0.00 |
| Given Training | Uniform Testing |
| Accuracy | 0.33 |
| Precisions (+, n, −) | 0.00, 0.33, 0.00 |
| Recalls (+, n, −) | 0.00, 1.00, 0.00 |
| $F_1$ Scores (+, n, −) | 0.00, 0.50, 0.00 |
| Uniform Training | Given Testing |
| Accuracy | 0.17 |
| Precisions (+, n, −) | 0.00, 0.00, 0.17 |
| Recalls (+, n, −) | 0.00, 0.00, 1.00 |
| $F_1$ Scores (+, n, −) | 0.00, 0.00, 0.29 |

Table 18: Baseline classifier (Classifier 4) metrics

## 4 Analysis

### 4.1 How $M_f$ modifies the models

The results from Section 3.1 highlight a pattern. For lower max feature values $M_f$, the top models are logistic regressions, whereas for $M_f \geq 1000$, the best are Bernoulli naive Bayes. This suggests that logistic regressions are more versatile with smaller dimensions of features than other classifiers. This is likely due to their optimization feature weights. A smaller feature space results in fewer possible local minima. With fewer local minima, it is likely that the true error rate minimum is found with the feature weights.

For $M_f \geq 1000$, the Bernoulli naive Bayes models have the highest bound for the right-sided 95% cross-validation scores with larger feature spaces. Most of the classifiers considered rely on solving an optimization problem to determine a support vector, or a set of weights. The naive Bayes models do rely on optimization, and are therefore more reliable with large feature spaces.

### 4.2 The best classifier/parameter configurations

The best classifier configurations with a large feature space are naive Bayes estimators, as shown in Section 3.2. The highest scoring model is the Bernoulli naive Bayes, with $\alpha = 1$ laplace smoothing and sentiment/class priors included.

The Classifiers 1, 2, and 3 all perform better than the baseline model. The only exception is in the recalls of the 0-R, which the baseline model has the best results for. However, this is misleading since the 0-R baseline classifier labels all instances with the same class, guarranteeing that one of the classes has no false negative values.

With the used feature set, Bernoulli naive Bayes classifiers are more applicable, since most features only appear in a tweet as binary events. This is reflected in the best possible model, while the multinomial model is still high accuracy relative to non-naive Bayes methods (Table 14).

The naive Bayes classifiers yield higher accuracies than the other considered classifiers, since they scale easily with the size of the feature space (see Section 4.1). To improve the accuracy of optimization-reliant classifiers, I would need to either start with an initial state (of weights or support vectors) that is known to be near the true error maximum, or to train the models multiple times with a large number of iterations.

Both of these options cannot be achieved with limited time, and with a large feature space as I am using.

The top models rely on laplace smoothing with $\alpha = 1$, implying that there is a need to account for potentially missing values. However, as the *alpha* for the smoothing increases further, the features present in the data are undermined in predictions, and the resulting models are less accurate.

### 4.3 The most sentiment distribution agnostic classifier

Since the `Test.csv` and `Train.csv` data are not guarranteed to follow the same sentiment distributions, the true best model is selected by comparing $F_1$ scores. As shown in Section 3.3, the sentiment distribution of the training and testing data influences the evaluation metrics of classifiers.

For the top 3 Classifiers Section 3.3, training on given data and testing on uniform data yields high precisions and low recalls for non-neutral sentiments, indicating that they are harder to confirm than neutral ones. Thus, training on the data with the given sentiment distribution is not preferrable if the true testing data does not follow the same sentiment distribution. In fact, training on uniform data yields better accuracy even if the distribution of sentiments in the test set is not uniform. The opposite is true for uniform training and given testing sets, which has low precisions, but high recalls for non-neutral sentiments. Therefore, the models perform better for non-neutral sentiments trained on a uniform set instead of the full dataset.

Of the top classifiers, the highest $F_1$ scores (accounting for both precision and recall) for a uniform training set come from the Bernoulli naive Bayes classifiers. Since both of these models have the same $F_1$ scores per sentiment, the true *best* model is determined otherwise.

The only hyperparameter that differs between Classifier 1 and Classifier 2 is inclusion of class priors. To consider how the hyperparameter affects the model, high precision and high recall (or simply the $F_1$ Scores) for non-neutral classes are compared, since neutral sentiment is common in the training set. Fitting the prior labels reduces the accuracy when training on the given set and testing on the uniform set. This is a small difference of 0.01 in accuracy. However, it confirms that if the `Test.csv` dataset has a different sentiment distribution to the `Train.csv`, it is preferrable to prevent prior distributions from influencing predictions.

## 5 Conclusions

Performing sentiment analysis on tweets is a difficult task. Vectorization of the features yields a large feature space, which contributes to the time and space complexity in training classifiers. While minimizing the feature space reduces complexity, it also sacrifices the accuracy of classifiers (some more than others).

Testing a multitude of different classifier combinations, the highest scoring (calculated by Section 2.4.1) classifiers are naive Bayes classifiers, with the best model relying specifically on a Bernoulli naive Bayes classifier. Bernoulli models make sense in this case, given the small sizes of tweets meaning features appear no more than once in an instance (Pardes, 2017). With little information on the `Test.csv` set, the models generated need to be highly accurate regardless of the distributions of the training sentiments.

**Of the classifiers considered, the most efficacious, is a Bernoulli naive Bayes model trained on the maximum subset of data with uniform sentiment distribution with laplace smoothing $\alpha = 1$, without fitting prior distributions.**

## References

Luciano Barbosa and Junlan Feng. 2010. Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 36–44, USA. Association for Computational Linguistics.

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *Processing*, 150, 01.

Arielle Pardes. 2017. A brief history of the ever-expanding tweet. *Wired*.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 502–518, Vancouver, Canada, August. Association for Computational Linguistics.

R. L Weide. 1998. Carnegie mellon pronouncing dictionary. www.cs.cmu.edu.