COMP30027 Assignment 2: Report

Anonymous

1 Introduction

Sentiment analysis is a field of natural language processing focusing on the classification of text by its perceived sentiment. It speeds up measuring the opinions of groups of people and can be applied in many fields such as marketing, politics or sociology. My goal is to develop and train three differenct reliable Twitter sentiment classifiers and one baseline model. This involves developing methods to extract and select useful features from a dataset of posts, then to choose and evaluate highly reliable classifier models.

1.1 Dataset

The given dataset provided contains two lists of Twitter posts (tweets) made on the platform prior to 2017 (Rosenthal et al., 2017). The two files enclosed are a `Train.csv` for training and a `Test.csv` for testing. Each tweet is an instance in the dataset. The training set contains 21802 labelled instances and the testing set contains 6099 instances. For each instance, included is the tweet text and tweet ID. The tweets included vary in content. For example, some tweets are not entirely in English: "*season in the sun versi nirvana rancak gak..slow rockkk...*". In the training file also contains the true sentiment of each tweet. Tweets can either have a "positive", "neutral" or "negative" sentiment, distributed as shown in Figure 1.

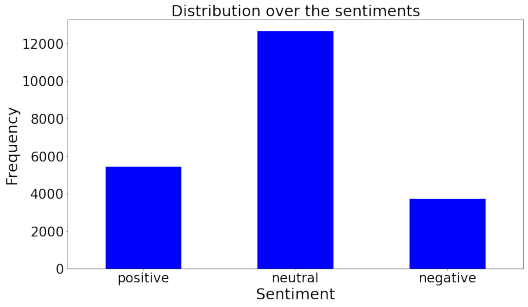

Figure 1: Distribution of the sentiments

2 Methodology

This section contains a breakdown of the process through which the final models are developed. The following methods are developed with reference to prior works on Twitter sentiment analysis by Go et al. (2009) and Barbosa and Feng (2010).

2.1 Instance cleaning

Some of the features extracted rely on the text in the tweets to be pruned of unwanted characters and words. I have opted to generate a separate list containing the cleaned versions of tweets. Cleaning involves removing stopwords (Section 2.1.1), links, tweet hashtags, tweet mentions, numbers, non-alphanumeric characters. Also performed is the reduction of repeated letters with more than two occurrences to just two, as suggested by Go et al. (2009).

Neither stemming nor lemmatization are used in the final models as I consider both too language-dependent to be useful in non-English instances.

2.1.1 Stopwords

Manual stopword list construction is tedious and inexhaustive (I cannot identify stopwords in certain languages). Therefore, I have opted to start with the Python Natural Language Toolkit's (`NLTK`) defined stopword list, which includes multiple languages (Bird et al., 2009). To test the efficacy of this set, I generate a set of word clouds over the training cleaned tweets (using the `NLTK` stopword list) grouped by sentiment. This then informs whether more terms need to be manually added the used list does not include.

2.2 Feature extraction

2.2.1 How many features per feature type is enough?

With over 20000 instances in the training dataset, some of the feature types defined will

generate a massive range of results. When vectorizing, it is best practice to implement a hard maximum for the unique number of features in a feature type. 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.

2.2.2 Twitter features

First, relying on personal usage experience with Twitter, I extract the main platform-specific features from the tweet text. These are:

- Hashtags (e.g. `#term`): Used to associate tweets with a certain term on the platform.
- Mentions (e.g. `@user`): Used when addressing a specific user on the platform.
- Links (e.g. `https://t.co/id`): Used to link to a website with an id on the platform's redirect service.

These are integrated within the Twitter platform, meaning they are widely used by users. Therefore, these features may be strongly correlated to the sentiments and should be isolated for use in the final models.

2.2.3 Linguistic features

Next, linguistic features are extracted and used to tokenize the tweet in different ways.

- Part-of-Speech Tags: Extracting the grammatical types of words.
- Words
- Word 2-Grams: Word pairs used in the tweet.
- Word Lengths: The lengths of words used as tokens.
- Phonetics: The phenomes of english words as defined in the CMU Pronounciation Dictionary (Weide, 1998).
- Alphabetic Characters
- Punctuation (`.?!,:;-()[]{}"'/`)

These feature types may all have features which are strongly correlated to sentiments. For example, a positive `big win` versus a negative `big disaster` word 2-gram.

- Emoticons (e.g. `:(` or `:)`): The rise of `ASCII` emoticons allows users to quickly express their emotions, which correlate strongly to the sentiment of their text.

Extracting emoticons is done differently to the method used in the Go et al. (2009) research, since their method misses a large number of emoticons, such as the surprised duck `:v` emoticon. Instead, emoticons are defined as a string comprised of eyes (`;:8=`), optional middle characters (`,'-"*`), and mouths divided into categories: happy (`)3]`), sad (`/([`), neutral (`pl|`) and surprised (`vo`). The detected emoticons (including reversed versions) are simplified into one of four emoticons based on their mouth category: happy `:)`, sad `:(`, neutral `:|` and surprised `:o`.

2.2.4 Metric features

These are largely numeric counts of the previously mentioned feature types within a tweet.

- Number of Words
- Number of Characters
- Number of Alphabetic Characters
- Number of Links
- Number of Hashtags
- Number of Mentions
- Number of Emoticons
- Quoting (e.g. `"text"`): Can also be called *retweets*, represented in the dataset with quote marks.
- Average Word Length

These metrics can help differentiate between a neutral tweet and a sentimental tweet (e.g. emoticons are correlated to emotion).

2.2.5 Vectorization

For all non-metric feature types, they can be vectorized by $TF\text{-}IDF$, or by occurrence counts. At the time that the data was collected, tweets were limited to 140 characters (Pardes, 2017). This suggests that most features (including word pairs) will only appear at most once in a tweet. Therefore, non-metric features are vectorized with their occurrence counts.

2.3 Classifier Selection

2.3.1 Baseline

The baseline model used will be 0-R: most frequent class. All chosen models need to outperform this one.

2.3.2 Classifiers Considered

The following classifiers and parameters are first all measured as described in Section 2.4.1:

Multinomial/Bernoulli Naive Bayes: Naive bayes classifiers each assuming likelihood distributions of features corresponding to their name.

Parameter	Values Considered
Include Label Priors	Yes, No
Alpha Smoothing (α)	0, 1, 10

Table 1: Naive Bayes Parameter Sets

Logistic Regressions: Linear statistical models for categorical labels.

Parameter	Values Considered
Fit Intercept	Yes, No
Maximum Iterations	100, 500
Optimizer Algorithm	`sag`, `saga`

Table 2: Logistic Regression Parameter Sets

Decision Trees: Branching tree classifiers.

Parameter	Values Considered
Maximum Depth	1, 100, 500

Table 3: Decision Tree Parameter Sets

K **Nearest Neighbours**: Decides classes based on the K closest neighbours' class labels (in a weighted voting system).

Parameter	Values Considered
K Neighbours	1, 10, 100, 500
Vote Weighting	`uniform`, `distance`

Table 4: K-Nearest Neighbour Parameter Sets

Support Vector Classifiers: Create borders between classes with the use of support vectors.

Parameter	Values Considered
Kernel Function	Linear, Cubic
Regularization C	0.1, 1, 3
Decision Function	One-v-One, One-v-Rest

Table 5: Support Vector Classifier Parameter Sets

2.4 Evaluation

2.4.1 Which Classifier/Parameter Configurations Are Best?

To reduce the inital set of possible estimators to just two selected candidates, they are each cross-validated twice. First, 5-fold cross validation scores are calculated on the dataset as is. Then, 5-fold cross validation scores are calculated on largest subset of the data with a uniform distribution of sentiments. These 10 scores are then averaged $\bar{x}$, and their standard deviations s found. The top candidate classifier/parameter configurations are found based on the highest approximate 80% lower bounds ($\bar{x} - s$).

2.4.2 Evaluation Metrics

The following metrics are generated for each classifier that makes it past Section 2.4.1:

Metric	Formula
Accuracy	$\frac{\#Correct}{n}$
Precision (per class)	$\frac{TP}{TP+FP}$
Recall (per class)	$\frac{TP}{TP+FN}$
F_1-Score (per class)	$\frac{2TP}{2TP+FP+FN}$

Table 6: Metrics calculated for chosen models

On top of this, the model's confusion matrix is also generated, for visual representations potential biases.

2.4.3 Which models are label distribution agnostic?

It is unknown whether the `Text.csv` labels are distributed similarly to those in `Train.csv`. Therefore evaluation metrics are generated on four $4:1$ train to test sets. First are generated a set of training and testing sets on the maximum subset of the data with uniform sentiment distribution. Next are generated a set of training and testing sets on the data with sentiments distributed as given (Randomly). From these, four sets of evaluation metrics are generated:

- Uniform Training and Random Testing
- Uniform Training and Uniform Testing
- Random Training and Random Testing
- Random Training and Uniform Testing

The evaluation metrics as generated in Section 2.4.2 are then compared to see which of the models (including the baseline) are most agnostic to sentiment distributions.

3 Results

3.1 Comparing the effects of stopword removal

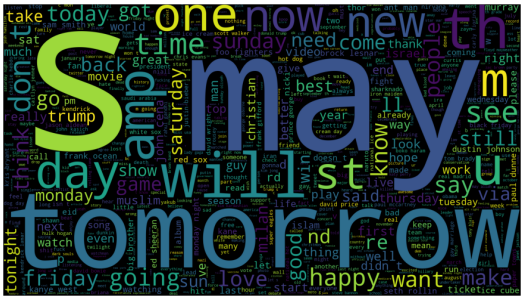

Figure 2: Word cloud over the cleaned tweets without stopword removal

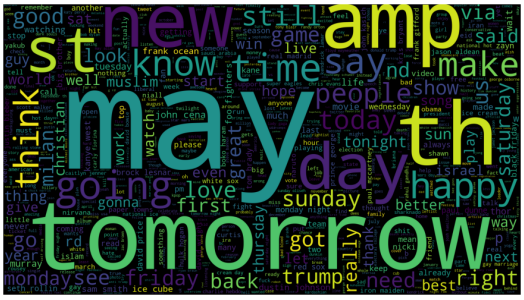

Figure 3: Word cloud over the cleaned tweets with removal using the `NLTK` list

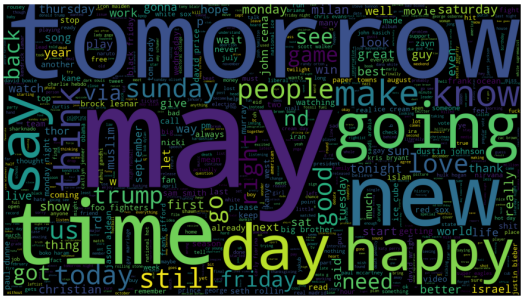

Figure 4: Word cloud over the cleaned tweets with removal using the modified `NLTK` list

3.2 Highest cross-validation accuracy per number of maximum features

After generating calssifiers for each of the following values M_f, the following boasted the highest scores calculated using Section 2.4.1.

Parameter	Value
Classifier	Logistic Regression
Fit Intercept	Yes
Maximum Iterations	500
Optimizer Algorithm	`sag`
Accuracy	0.5529
Standard Deviation	0.04754

Table 7: Top classifier for $M_f = 10$

Parameter	Value
Classifier	Logistic Regression
Fit Intercept	Yes
Maximum Iterations	500
Optimizer Algorithm	`sag`
Accuracy	0.5858
Standard Deviation	0.03109

Table 8: Top classifier for $M_f = 100$

Parameter	Value
Classifier	Bernoulli NB
Include Priors	Yes
α Smoothing	1
Accuracy	0.6054
Standard Deviation	0.01067

Table 9: Top classifier for $M_f = 1000$

Parameter	Value
Classifier	Bernoulli NB
Include Priors	Yes
α Smoothing	1
Accuracy	0.6321
Standard Deviation	0.01504

Table 10: Top classifier for $M_f = 5000$

Parameter	Value
Classifier	Bernoulli NB
Include Priors	Yes
α Smoothing	1
Accuracy	0.6329
Standard Deviation	0.01237

Table 11: Top classifier for $M_f = 10000$

Since the model accuracy begins to plateau at $M_f = 10000$, The final classifer models are con-

structed using $M_f = 10000$ as an upper bound for features per feature type.

3.3 The top 3 parameter/classifier configurations

The following parameter configurations yield the 3 highest scores as defined in Section 2.4.1

Parameter	Value
Classifier	Bernoulli NB
Include Priors	Yes
α Smoothing	1
Accuracy	0.6329
Standard Deviation	0.01237

Table 12: Top classifier (Classifier 1) for $M_f = 10000$

Parameter	Value
Classifier	Bernoulli NB
Include Priors	No
α Smoothing	1
Accuracy	0.6319
Standard Deviation	0.01210

Table 13: 2nd best classifier (Classifier 2) for $M_f = 10000$

Parameter	Value
Classifier	Multinomial NB
Include Priors	Yes
α Smoothing	1
Accuracy	0.6319
Standard Deviation	0.01210

Table 14: 3rd best classifier (Classifier 3) for $M_f = 10000$

It appears that the top classifiers according to the used scoring value are all Naive Bayes classifiers.

3.4 Generating evaluation metrics for the final 4 models

Generated based on Section 2.4.2. Per-class metrics are shown in order of postive, neutral, then negative sentiments.

Parameter	Value
Classifier	0-R
Training	Uniform
Testing	Uniform
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
Training	Random
Testing	Random
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
Training	Random
Testing	Uniform
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
Training	Uniform
Testing	Random
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 15: Metrics of Baseline classifier for $M_f = 10000$

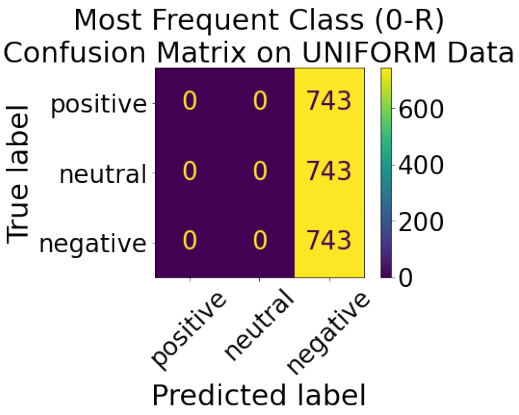

Figure 5: Baseline classifier's confusion matrix for uniform training and uniform testing data

4 Analysis

4.1 How the stopword removal modifies the data

The cleaning technique used highlights a large number of unformed word parts, such as "s" or "u", as shown in Figure 2. The initial stop-

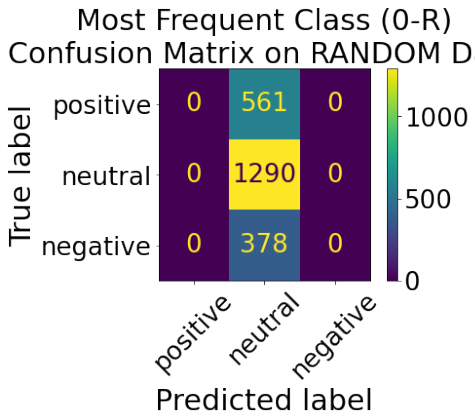

Figure 6: Baseline classifier's confusion matrix for random training and random testing data

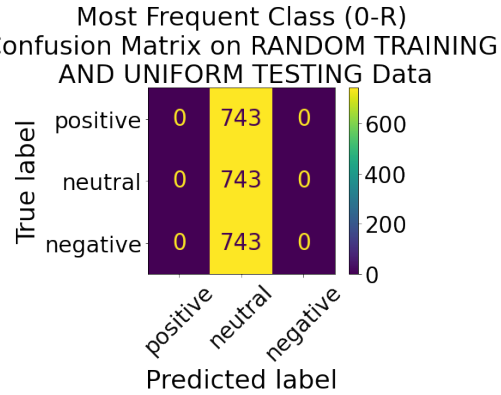

Figure 7: Baseline classifier's confusion matrix for random training and uniform testing data

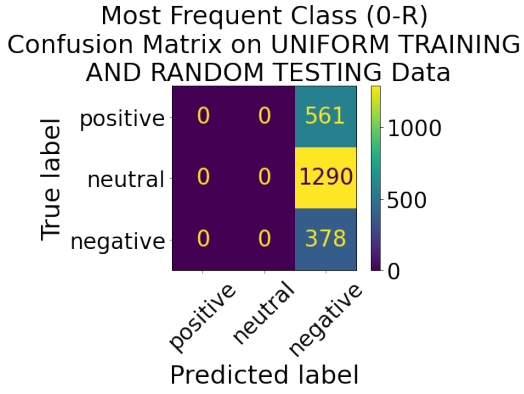

Figure 8: Baseline classifier's confusion matrix for uniform training and random testing data

word list misses some highly repeated terms with no meaning (e.g. "*th*") as is displayed in Figure 3. These terms need to be removed manually, to yield a final word cloud shown in Figure 4. Although words like "*tomorrow*" appear frequently, they may be parts of sentiment indicative word 2-grams (e.g. "*happy tomorrow*"), and are therefore not removed.

Parameter	Value
Classifier	Bernoulli NB
Include Priors	Yes
α Smoothing	1
Training	Uniform
Testing	Uniform
Accuracy	0.62
Precisions (+, n, –)	0.65, 0.54, 0.64
Recalls (+, n, –)	0.69, 0.43, 0.73
F_1 Scores (+, n, –)	0.67, 0.48, 0.68
Training	Random
Testing	Random
Accuracy	0.61
Precisions (+, n, –)	0.78, 0.60, 0.50
Recalls (+, n, –)	0.14, 0.98, 0.74
F_1 Scores (+, n, –)	0.24, 0.74, 0.01
Training	Random
Testing	Uniform
Accuracy	0.41
Precisions (+, n, –)	0.94, 0.36, 1.00
Recalls (+, n, –)	0.21, 0.99, 0.02
F_1 Scores (+, n, –)	0.35, 0.53, 0.04
Training	Uniform
Testing	Random
Accuracy	0.64
Precisions (+, n, –)	0.56, 0.87, 0.48
Recalls (+, n, –)	0.81, 0.51, 0.84
F_1 Scores (+, n, –)	0.66, 0.64, 0.61

Table 16: Metrics of Classifier 1 for $M_f = 10000$

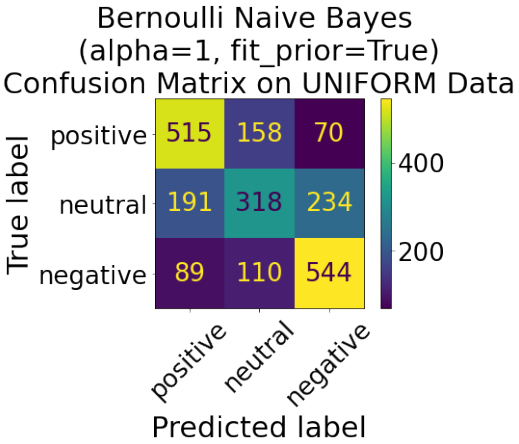

Figure 9: Classifier 1's confusion matrix for uniform training and uniform testing data

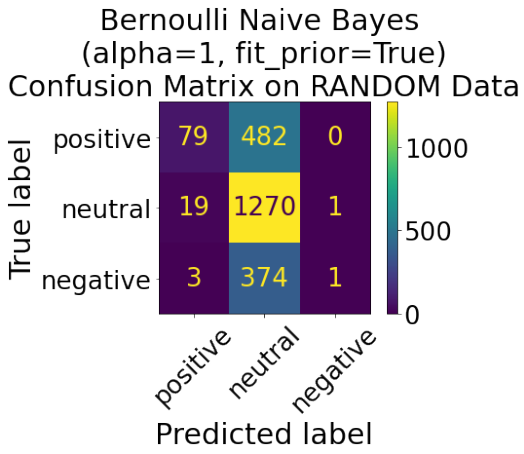

Figure 10: Classifier 1's confusion matrix for random training and random testing data

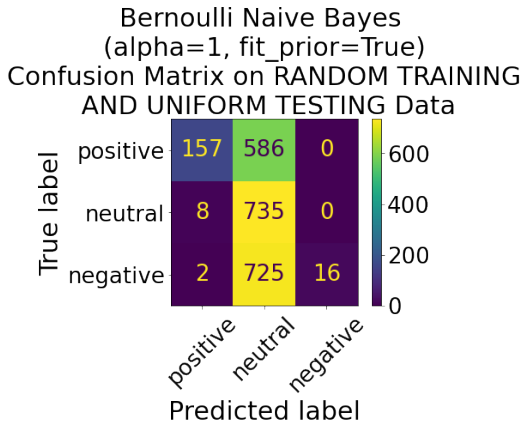

Figure 11: Classifier 1's confusion matrix for random training and uniform testing data

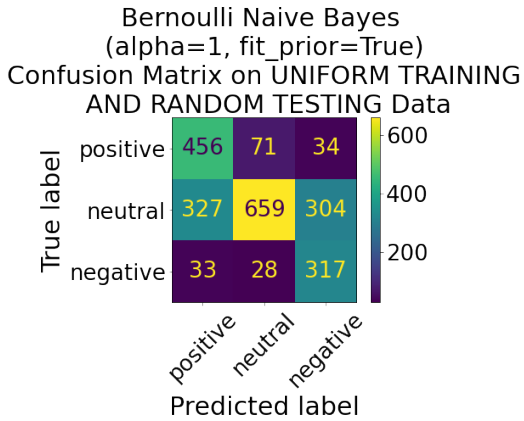

Figure 12: Classifier 1's confusion matrix for uniform training and random testing data

4.2 How the maximum number of features per feature type M_f modifies the models

The results shown in Section 3.2 highlight an interesting pattern in the classifiers. For lower

Parameter	Value
Classifier	Bernoulli NB
Include Priors	No
α Smoothing	1
Training	Uniform
Testing	Uniform
Accuracy	0.62
Precisions (+, n, –)	0.65, 0.54, 0.64
Recalls (+, n, –)	0.69, 0.43, 0.73
F_1 Scores (+, n, –)	0.67, 0.48, 0.68
Training	Random
Testing	Random
Accuracy	0.61
Precisions (+, n, –)	0.74, 0.60, 0.75
Recalls (+, n, –)	0.17, 0.98, 0.01
F_1 Scores (+, n, –)	0.27, 0.74, 0.02
Training	Random
Testing	Uniform
Accuracy	0.42
Precisions (+, n, –)	0.91, 0.36, 0.96
Recalls (+, n, –)	0.25, 0.98, 0.04
F_1 Scores (+, n, –)	0.39, 0.53, 0.07
Training	Uniform
Testing	Random
Accuracy	0.64
Precisions (+, n, –)	0.56, 0.87, 0.48
Recalls (+, n, –)	0.81, 0.51, 0.84
F_1 Scores (+, n, –)	0.66, 0.64, 0.61

Table 17: Metrics of Classifer 2 for $M_f = 10000$

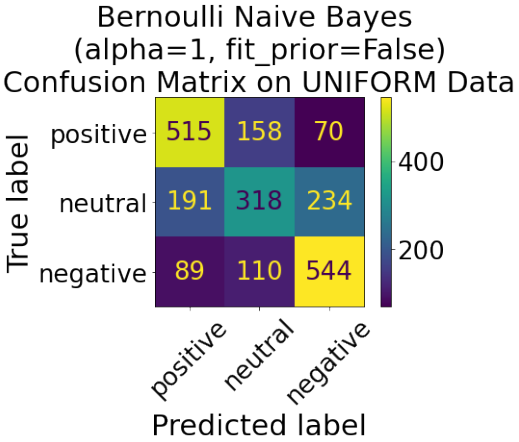

Figure 13: Classifier 2's confusion matrix for uniform training and uniform testing data

max feature values M_f, the highest performing model is the logistic regression, whereas past $M_f = 1000$, the best model is the Bernoulli Naive Bayes. This suggests that logistic regres-

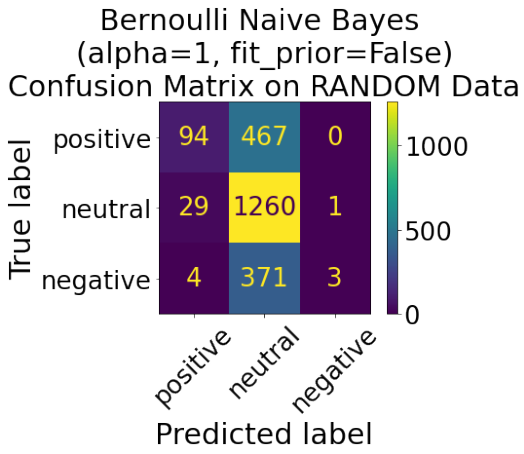

Figure 14: Classifier 2's confusion matrix for random training and random testing data

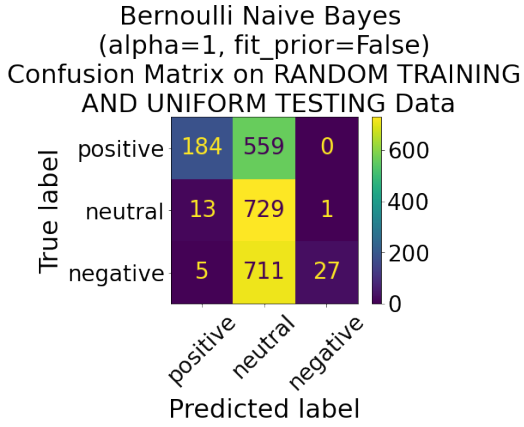

Figure 15: Classifier 2's confusion matrix for random training and uniform testing data

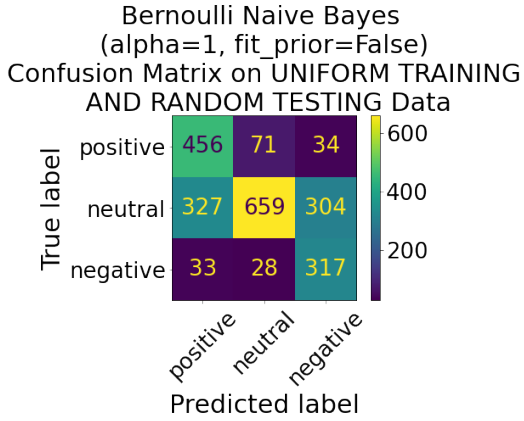

Figure 16: Classifier 2's confusion matrix for uniform training and random testing data

sions are better for smaller dimensions of features. This is likely due to the logistic regressions reliance on optimization to find the best

Parameter	Value
Classifier	Multinomial NB
Include Priors	Yes
α Smoothing	1
Training	Uniform
Testing	Uniform
Accuracy	0.62
Precisions (+, n, –)	0.66, 0.57, 0.62
Recalls (+, n, –)	0.66, 0.42, 0.78
F_1 Scores (+, n, –)	0.66, 0.48, 0.69
Training	Random
Testing	Random
Accuracy	0.61
Precisions (+, n, –)	0.76, 0.60, 0.60
Recalls (+, n, –)	0.19, 0.97, 0.01
F_1 Scores (+, n, –)	0.30, 0.75, 0.02
Training	Random
Testing	Uniform
Accuracy	0.42
Precisions (+, n, –)	0.92, 0.36, 1.00
Recalls (+, n, –)	0.26, 0.98, 0.02
F_1 Scores (+, n, –)	0.41, 0.53, 0.04
Training	Uniform
Testing	Random
Accuracy	0.62
Precisions (+, n, –)	0.57, 0.87, 0.44
Recalls (+, n, –)	0.78, 0.48, 0.87
F_1 Scores (+, n, –)	0.66, 0.62, 0.59

Table 18: Metrics of Classifer 3 for $M_f = 10000$

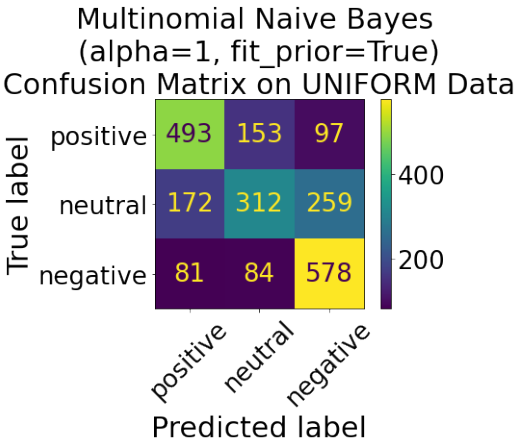

Figure 17: Classifier 3's confusion matrix for uniform training and uniform testing data

model. Since optimization is perfomed to find local minima in errors in the model, a smaller feature space results in fewer local minima, and therefore a higher likelihood in the local mini-

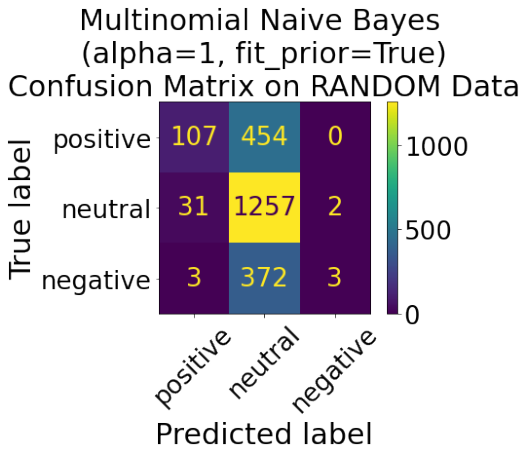

Figure 18: Classifier 3's confusion matrix for random training and random testing data

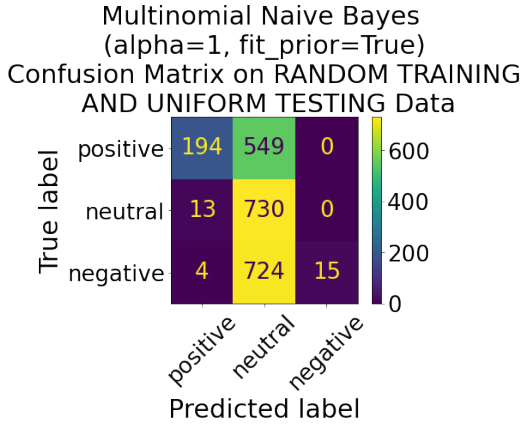

Figure 19: Classifier 3's confusion matrix for random training and uniform testing data

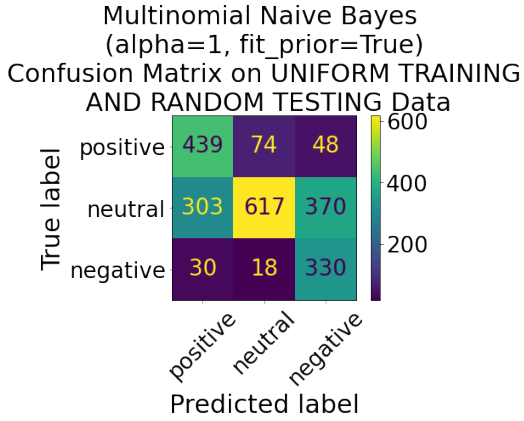

Figure 20: Classifier 3's confusion matrix for uniform training and random testing data

mum being the overall minimum of errors. The results also highlight the Bernoulli Naive Bayes model, as it yields the highest 80% accuracy for 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 in complex feature spaces.

4.3 The best classifier/parameter configurations

The three best model configurations for a large feature space are Naive Bayes estimators, as shown in Section 3.3. The highest scoring model is the Bernoulli Naive Bayes, with $\alpha = 1$ laplace smoothing and sentiment label priors included.

Both multinomial and bernoulli naive bayes classifiers are very similar in function, one relies on multiple possible events, whereas the other relies on binary events. With this data and feature set, the bernoulli naive bayes classifier tends to be more applicable, since most features either appear or don't appear within a tweet (appear in 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 tend to yield higher accuracies than the other considered classifiers, since they scale easily with the size of the feature space. Other models rely on optimization, which increases in complexity with the feature space. As more features are added, there will be a higher number of possible minima and maxima to consider when optimizing, decreasing the chances of the true best configuration being discovered. I would need to either start with an initial state that is known to be closest to the most accurate, or to train the models on a large number of iterations. Both of these options are not realistic with limited time, and with a large feature space as I am using.

4.4 Of the final 4 models, which are most sentiment distribution agnostic

5 Conclusions

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.