Text Mining Assignment 3
Comparing Naïve Bayes and LSTM networks for Sentiment Analysis of tweets

I-fan Lin s2814412
Laura Jansen-Storbacka s2691116

December 2021-January 2022

Introduction

Sentiment analysis involves analysing texts and classifying them according to the feelings or sentiments of the author. Together with topic identification, this can be used to gauge either individual feelings or public sentiment about a particular topic. Together with topic modelling, sentiment analysis can be used to understand how people feel about a particular subject. This can be useful in areas such as politics, as well as in business and marketing. Twitter data consists of tweets, short phrases or sentences less than 280 characters in total. Tweets can potentially be used to capture the zeitgeist, getting a snapshot of the public mood in regard to a topic. This short informal format means the data is abundant, making it an ideal task for text mining techniques.

This assignment attempts SubTask 4A from SemEval-2017: Sentiment Analysis in twitter. SubTask 4 is defined as " Message Polarity Classification: Given a message, classify whether the message is of positive, negative, or neutral sentiment". Happy and enthusiastic tweets should be classified as positive , factual statements or tweets without emotion should be classified as neutral, and sad, angry or complaining tweets should be classified as negative.

Sentiment analysis models

There are two main approaches to sentiment analysis, rule based systems and machine learning models. Rule based systems are prescriptive, and provide lists of positive and negative words and phrases. The problem is that language is flexible and dynamic, and rules make it hard to detect subtleties such as irony and sarcasm where words suddenly change their meanings. Supervised machine learning models on the other hand require only labeled data, and learn classification rules from a training set. Depending on the model this may require a large volume of data to work well, and can be computationally heavy. Sentiment analysis can be performed using shallow machine learning models e.g. Naïve Bayes, Support Vector Machines or Random Forests. It can also be performed using deep neural networks, in particular Long Short-Term Memory (LSTM) models, that are able to consider context and word order. Transformer models are the state-of-the art models which involve using large pre-trained language models which are then fine tuned for a specific situation, saving a lot of time and computing power. There are also hybrid systems that combine rule based and machine learning systems, for example using lexicons of positive and negative words can use rules to augment the features of a machine learning model.

Data Description

The data consists of tweets collected from twitter collected between 2013 and 2016. The tweets were selected to represent different current topics. Tweets were collected in Arabic and English, but for this task we used only English tweets. All tweets had been human annotated with a sentiment rating (positive, neutral or negative).

Class	Total
Neutral	22591
Positive	19903
Negative	7840

Table 1: *Tweets by sentiment*

In the initial SemEval task the topics of the test and train sets were different, this tested the ability of a model to adapt to a new topic. For our analysis however we merged the twitter datasets from the SemEval competitions from $2013 - 2017$ into a large combined set of 50334 tweets. The combined set was then split into test and train. In contrast to the original SemEval our train and set each contained a variety of tweet types. This gave us a larger and more diverse dataset. Initial analysis of the combined dataset showed that the classes were quite unbalanced with the negative class being much smaller than the positive or neutral classes. Unbalanced classes caries a risk of increased bias against the smaller class as the incorrect classification of this class carries a relatively smaller penalty (Hastie et. al. 2001). We hypothesized that any increase in bias caused by the unbalanced classes can be offset because of the large size of our dataset and the consequent reduction in variance. The combined dataset consisted of an array of tweets. For every tweet there were 3 parts; The twitter ID (this was not relevant to our research), the sentiment tag (positive, negative or neutral), and the text of the tweet.

Research Question

The original SemEval task 4A is to classify tweets into positive, negative and neutral. In this assignment we compare the performance on this task of a shallow learning algorithm Categorical Naïve Bayes, with the performance of a deep learning LSTM network. Both were chosen as they have both been shown to perform well with sentiment classification tasks (Jurafsky and Martin, 4.4). We extend the dataset by combining the tweets from 2013-2017 into one document. The dataset was split 80:20 into training and test data, and the same split was used for all comparisons and for both model types.

Pre-Processing

The high dimensionality of the dataset (over $70,000$ unique tokens) posed processing challenges for both Naive Bayes and LSTM. We used various dimension reduction techniques to reduce the total. The tweets were tokenized, and this involved removing all punctuation. Although some punctuation, especially "!" contributes to sentiment, we did not consider that it is either positive or neutral. All words were made lower case, and lemmatization was used to further reduce the vocabulary. The stop word list was edited to ensure negative words such as "not", "can't", and "'t" which contain a lot of information about sentiment were still included in the vocabulary. Additionally twitter handles beginning with "@" and URLs were removed from the sentences.

We initially ran the models with all the possible tokens, after the pre-processing described above this was around $50,000$ words. We were interested in whether removing less

frequent words improved model performance, so we reduced the unique tokens to include only words in the vocabulary that occurred more than once. The theory was that a rare word that only occurs once cannot be very useful as a predictor. This reduced the vocabulary size to a more manageable $17,841$ words.

Pre-Processing for Naïve Bayes: Bag Of Words

Naïve Bayes required a bag of words model. This meant that all the sentences were represented as feature vectors across the entire vocabulary. Each sentence was a vector the length of the total vocabulary, with 1 if a word occurred 1 or more times in the sentence, and 0 if it did not. We initially ran the model with all the possible tokens, after the pre-processing described above this was 43161 words. Later we reduced this to include only words in the vocabulary that occurred more than once. The theory was that a rare word that only occurs once cannot be very useful as a predictor. This reduced the vocabulary size to a more manageable $17,841$ words.

Pre-processing for LSTM: Word Embeddings

Word embedding refers to vector representations of words, and it is is based on an idea called the distributional hypothesis that words in the same context often share similar meanings. The dimension of the word vectors is a hyperparameter.

In our assignment, we use static[1] word embeddings, and explore the effects of different dimension of embeddings. The word embeddings were derived in two different ways, with the **word2vec** model and using an embedding layer in a neural network. The **word2vec** model is a form of unsupervised learning, whereas the neural network with embedding layer is supervised learning.

Naïve Bayes

Naïve Bayes is called naïve because it has the assumption that all features are independent. In text data where words are organised in sentences this is obviously not true as words gain meaning from their neighbors, however, the model is still known to perform well for sentiment analysis. We use binary Naive Bayes where the presence or absence of a word in a document or tweet is indicated by a 1 or a 0, and if the same word occurs more than once this is ignored.

Long Short-term Memory (LSTM)

LSTM is a kind of deep learning architecture that is particularly useful for the sequence data. It is an improved and modified version of a Recurrent Neural Network (RNN), as RNN suffer from the short memory and can not deal with long sequence inputs.

LSTMs have a relatively longer memory because they have a design of three gates: the input gate, the forget gate and the output gate. Each gate is a Sigmoid function, and for each function there are two parameter tensors as input. The model learns these parameter tensors. The following formula [2] shows that the input gate and the forget gate decide the long-term state vector.

[1] The word embeddings can be static and dynamic. For example, the two algorithms in **word2vec** produce fixed word embeddings from text, while the word embeddings in a Bert model will vary according to context.

[2] The formula is from Géron, A. (2019)

$$c_t = f_t \otimes c_{t-1} + i_t \otimes g_t$$

c_t means the long term state vector in t_{th} time step/character; f_t is forget gate in t_{th} time step/character; i_t is input gate in t_{th} time step/character; g_t is the vanilla RNN output. $\otimes$ is element-wise multiplication.

In this section, various combinations of architectures and embeddings will be tried (9 models in total), and the performance will be evaluated based on accuracy rate of validation data, which account for 12.5% of training dataset. The best models for full dataset and reduced dataset, respectively, will be used to compare with other machine learning methods.

Hyperparameters for LSTM

To find the best hyperparameters for the LSTM model, a validation set consisting of 12.5% of the training data was set aside. The hyperparameters of models 1 to 4 can be seen in table 2, and the hyperparameters of models 5 to 11 in table 3. Models 1 to 4 used word embeddings generated from **word2v** as input. They only differ in the depth of hidden layers and the data set they used. Models 3 to 9 use encoded words as input, the word embeddings are learnt during training. They differ in the dimensions of the word embeddings, the depth of hidden layer, and the data set they used.

Table 2: *Word Embeddings As Input: Hyperparmeters*

(a) *Models (Unit: %)*

	Model 1	Model 2
Hidden layer	(256,64)	(512,256,64)
Using redused dataset	No	No
	Model 3	Model 4
Hidden layer	(256,64)	(512,256,64)
Using redused dataset	Yes	Yes

(b) *The Model Hyperparameter And Architecture*

Input layer	
Input shape	$60 \cdot 100$
Hidden layer	
Kernel initializer	Glorot uniform
Activation	tanh
Drop out	None
Output layer	
Output activation	Softmax
Output shape	$1 \cdot 3$
Optimizer	Adam ($lr = 0.001$)
Loss function	Sparse Categorical Crossentropy
Early stopping applied	Yes

Table 3: *Words Embedding As Hidden layer: Hyperparmeters*

(a) *Models (Unit: %)*

	Model 5	Model 6	Model 7
Embedding dimension	300	100	50
Hidden layer	(512,64)	(512,256,64)	(256,64)
Using redused dataset	No	No	No
	Model 8	Model 9	Model 10
Embedding dimension	300	100	50
Hidden layer	(512,64)	(512,256,64)	(256,64)
Using redused dataset	Yes	Yes	Yes

(b) *The Model Hyperparameter And Architecture*

Input layer	
Input shape	$None$
Hidden layer	
Kernel initializer	Glorot uniform
Activation	tanh
Drop out	None
Output layer	
Output activation	Softmax
Output shape	$1 \cdot 3$
Optimizer	Adam ($lr = 0.001$)
Loss function	Sparse Categorical Crossentropy
Maximum epochs	30
Early stopping applied	Yes

Results From Hyper-Parameter Search

The validation accuracy can be seen in 4. Based on the results from the 11 models, we found that models trained on the reduced dataset perform better than those trained on the full dataset. For example, model 1 vs model 3, model 2 vs model 4, model 5 vs model 8, and so on. This confirms that words with frequency 1 are noisy data that negatively impact model learning. Additionally, models with the word embeddings created using supervised learning performed better than those with embeddings created using unsupervised learning. For example, model 2 and model 6 have the same embedding dimension of 100, but the accuracy rate of model 2 is 59.22%, while for model 6 it is 64.32%. The optimal model trained with the full dataset was model 7, and the optimal model trained with the reduced dataset was model 8. These two models were used in the comparison with Naïve Bayes.

Table 4: *Validation Accuracy Rate Of Models (Unit: %)*

	Model 1	Model 2	Model 3	Model 4	Model 5
Validation Accuracy Rate	59.10	59.22	59.54	60.27	64.76
	Model 6	Model 7	Model 8	Model 9	Model 10
Validation Accuracy Rate	64.32	65.28	65,85	65.24	65.40

Results

The results from the Naive Bayes models can be seen in table 5. We see that the accuracy for the Naïve Bayes model is not high, at only 0.62, this is the same for both models. We also see that the recall is worse for the Negative class. This confirms that when there are unbalanced classes the model becomes biased against the smaller class. For the larger model with the full dataset, the recall shows that only 33% of the truly negative reviews were correctly identified as such. In the reduced dataset where we had removed all the words that only occurred once, we saw a slight improvement; this model correctly identified 45% of the negative class in the test set. Comparison with the training set (for the smaller vocabulary set) showed that it is possible our model is over fitting, as the training accuracy was 11% higher at 0.73. and the training recall for the negative class was a lot better at 61%.

The results from the LSTM can be seen in table 6. We see that the accuracy for the

	NB full dataset (test)			NB reduced dataset (test)			NB reduced dataset (train)		
	Precision	Recall	F1-score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
Negative class	0.56	0.32	0.41	0.54	0.45	0.49	0.68	0.61	0.64
Neutral class	0.62	0.63	0.62	0.64	0.59	0.61	0.76	0.70	0.73
Positive class	0.62	0.72	0.67	0.62	0.72	0.67	071	0.81	0.76
Accuracy			0.62			0.62			0.73

Table 5: *Naïve Bayes classification results on test data for full (∼50,000 words vocabulary) and reduced (∼17,000 words vocabulary, unique words reduced) datasets, and on training data for the reduced dataset.*

LSTMs model is also not high, at only 0.65[3], and is also slightly higher than the Naïve Bayes model. The recall for negative class is also worst for the LSTM. Even when trained with the reduced dataset, the recall only improved slightly from 0.44% to 0.45%.

	LSTM full dataset from model 7 (test)			LSTM reduced dataset from model 8 (test)		
	Precision	Recall	F1-score	Precision	Recall	F1-Score
Negative class	0.54	0.44	0.48	0.55	0.45	0.49
Neutral class	0.62	0.80	0.70	0.64	0.73	0.68
Positive class	0.76	0.56	0.64	0.70	0.64	0.67
Accuracy over all classes			0.65			0.65

Table 6: *LSTM classification results on full (∼50,000 words vocabulary) and reduced (∼17,000 words vocabulary, unique words reduced) datasets.*

Discussion and Conclusion

The performance of our Naïve Bayes model was better than chance, but not as good as we had hoped. One possible improvement to our Naïve Bayes model could involve using lexicons of positive words (a hybrid model) , and adding a feature if the words were included in the tweets or not. Another possible improvement would be to use more refined parsing techniques to deal with finding negation. It is also possible that our model did not pick up on the use of emoji well enough and missed important information this way.

The LSTM models performed slightly better, with accuracy measures of 0.65 compared to the Naïve Bayes accuracy of 0.62, however the recall for the smaller negative class continued to be worse with recall scores of 0.48 for the larger model and 0.45 for the smaller. This again confirms that the imbalance in the classes is causing the model to be biased against the smaller class. A possible way to improve this would be to create a randomly sampled smaller dataset with balanced classes, with tweets in the negative class having a greater probability of selection.

Finally we note that successful sentiment analysis is not a straightforward task, and even human annotators do not always agree. It may not be possible to successfully classify all tweets, but even a reasonable proportion such as we have found here could be enough to give provide valuable and useful information, for example when combined with topic modelling, sentiment analysis can tell how negative or positive sentiments towards a

[3]The test accuracy is similar to validation accuracy, which implies the validation set is representative of the data.

topic change over time, the direction of change should be correct even if the actual numbers are not.

1 References

Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media.

Hastie, T., Tibshirani, R., Friedman, J. H. (2001). The elements of statistical learning: Data mining, inference, and prediction. New York: Springer

Jurafsky, D. Martin, J. (2020). Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition. 3rd Edition draft.

SemEval-2017 Task 4: Sentiment Analysis in Twitter