Deriving Sentiments from English Texts for the Creation of Digital Collages

Tianen Liu

April 29, 2020

Abstract

Sentiment analysis aims to predict emotions in texts using machine learning. Usually, such predictors require a large training dataset containing structured texts of a language. Assembling annotated datasets of sentiments is time-intensive and labor-intensive. Therefore, the main objective of this project was to determine if machine learning algorithms trained with movie reviews could be used to predict sentiments in texts drawn from other domains. We trained sentiment predictors using three machine learning algorithms: Naïve Bayes, Turney Algorithm, and Long Short-term Memory (LSTM) network. Our results show that the LSTM network exhibits the highest cross validation accuracy (86.17%), followed by Naïve Bayes (83.73%) and Turney Algorithm (50.87%). Our results also show that predicting sentiments in a different domain is challenging. When we train the model using movie reviews and test it on a dataset of status messages ("tweets") from Twitter, the accuracy for these three models drops to 52.09%, 53.48%, 37.68%, respectively. Thus, future work will focus on optimizing the LSTM network model to achieve better performance in classifying sentiments in a variety of texts.

Contents

List of Figures

List of Tables

1 Introduction

Sentiment analysis is a branch of natural language processing (NLP) that deals with the extraction of the semantic meaning from natural language data such as texts and speech. Widely used applications of NLP technologies include voice-controlled virtual assistants, spam email detection, YouTube auto subtitles, and so on. Sentiment analysis specifically extracts and classifies emotions in spoken and written language. One of the important applications of sentiment analysis is in psychiatry where it can help doctors and psychologists find imperceptible emotions of patients.

To apply a trained sentiment analysis model, a dataset of sentiments drawn from the application domain is often needed. For example, to train sentiment analysis models for psychiatry, a dataset of psychiatric texts is needed, comprising a text version of patients' descriptions of their mental conditions during psychiatric treatment sessions. However, collecting data for each specific application domain is challenging. For example, creating a novel dataset of sentiments requires taxing manual labeling of texts. Additionally, highly protected confidentiality may also hinder data collection.

To date, several datasets suitable for sentiment analysis modeling have been collected; they mostly focused on customer reviews. Examples include *Amazon review* (Ni et al., 2019), *IMDb movie review* (Maas et al., 2011), *Sentiment140* (Go et al., 2009), *Multi-domain sentiment dataset* (Blitzer et al., 2007). It is, therefore, interesting to examine how well models trained on user's reaction to movies, news, products can capture sentiments in other texts, such as novels, for example.

Therefore, the main objective of this research was to understand whether a model trained with a dataset from one domain could be used to predict sentiments in another dataset collected in a different context. There were three specific aims for this research.

1. Train three machine learning algorithms with a dataset of movie reviews for predicting positive and negative sentiments.

2. Compare the performance of these trained models.

3. Test their performance with datasets from different domains.

The overarching purpose of this research is to train an optimal model as a the foundation for *text2collage* (Kannan and Khuri, 2018), an automated tool that extracts images based on text descriptions and creates a digital collage. While some collages may be constructed for entertainment, others may be used to help patients visualize their emotions. Thus, having a model trained with sentiments found in consumer reviews that is applicable to extract emotions in other texts would be valuable. Hence, we evaluate the three models in terms of their applicability for embedding into the *text2collage* tool.

The remainder of this report is organized as follows. Prior work involving the three selected algorithms is reviewed in Section 2. Data acquisition and pre-processing, and detailed explanation of each of the three machine learning algorithms are described in Section 3. The experimental results are presented in Section 4, followed by the discussion of broader contribution of our research in Section 5. Finally, in Section 6, we make a conclusion about our findings and propose directions for future extensions of this work.

2 Prior Works

In NLP, sentiment analysis has been extensively studied in different domains. In this section, the focus is on a very small sample of prior research. For example, a comparison of three machine learning classifiers, including Support Vector Machine, Naïve Bayes, and Decision tree with the Turney Algorithm was performed on an Amazon dataset containing 2,500 positive and 2,500 negative reviews of commodities purchased on Amazon.com (Kanna and Pandiaraja, 2019). To train a Turney classifier, the data was pre-processed using part-of-speech tagging, two-word phrase extraction, seed words selection. Then the average semantic orientation (SO) of a review was computed. Each review is classified as positive, if the average SO is positive, and as negative if the average SO is negative.

The seed words include *good*, *excellent*, *best*, *great* for positive and *bad*, *poor*, *suck*, *terrible*, *horrible* for negative. Additionally, the *near* operator was introduced that computes the SO queries from the AltaVista

Advanced Search engine[1] to get the number of hits satisfying a condition that a particular two-word phrase is within the ten-word range of a seed word. All classification methods were compared using six metrics: true positive rate, false positive rate, precision, recall, F1-measure, and receiver operating characteristics (ROC). Results showed that Turney Algorithm yielded the highest accuracy (87.1%), followed by Naïve Bayes (87.0%), Support Vector Machine (69.6%), and Decision tree (61.1%), however, the difference in the performance of the Turney Algorithm and Naïve Bayes was negligible.

In addition to classical or shallow classifiers, artificial neural networks have also been used to predict sentiments in unseen texts. For example, a sentiment analysis of commodity reviews was done using *Word2vec*, by first converting texts into space vectors and then training a Multilayer Long Short-term Memory (LSTM) network (Sun et al., 2019). Compared to other data representations, *Word2vec* contained more contextual semantic information and helped with neural network training. The LSTM Network was implemented in Keras environment with parameters set as following: *binary_crossentropy* for loss, *RMSprop* for optimizer, 30 for the number of epochs, and 0.2 for dropout. Network optimizer and dropout rate were selected based on the accuracy outcomes.

Through experimentation, it was determined that through the use of dropout, the risk of over-fitting is effectively reduced. Over-fitting refers to the case when the model performs well while training, and fails to achieve a high performance on testing data or on different data. Dropout is a technique that randomly ignores ("drops out") some units from the neural network during training, which makes the layer look like it loses connections with the prior layer. This feature prevents the layers from co-adapting too much (Srivastava et al., 2014) and reduces over-fitting. The trained model resulted in a training accuracy of 99% and testing accuracy of 92%. By comparing with other machine learning methods, an LSTM network with double layers resulted in the highest accuracy (92%), followed by an LSTM network with single layer (91%), 1-Dimensional Convolutional Neural Network (1DCNN) (90%), and Support Vector Machine (86%).

A different architecture of the neural network addressed problems of vanishing and exploding gradients in Recurrent Neural Networks (RNNs) during model's training for sentiment analysis. As a result, several variants of the Gated Recurrent Neural Network were constructed and tested, namely the Gated Recurrent Unit (GRU), Bidirectional LSTM (Bi-LSTM) and Bidirectional GRU (Bi-GRU). By passing the input backwards, in addition to forward direction, Bi-LSTM and Bi-GRU can better preserve information. This feature enables networks to recognize long-term dependencies and contextual connections from both previous and future hidden states. Experimentation was conducted using the Amazon dataset of product reviews containing 120,000 training reviews, 40,000 *positive*, 40,000 *negative*, and 40,000 *neutral* reviews. Testing data contained 30,000 reviews (Sachin et al., 2020). Bi-GRU had the best performance measured by the accuracy (71%), GRU had the highest precision (71%), recall (71%), and F1 score (71%).

We next describe the datasets and methods used in our project.

3 Data and Methods

The overall computational workflow in this project comprises several steps, namely, (1) data acquisition and pre-processing, (2) model training and validation, and (3) the prediction of sentiments for unseen datasets.

3.1 Data Acquisition and Pre-processing

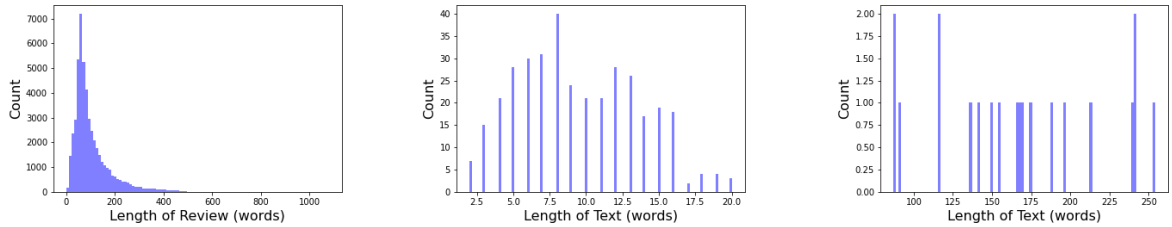
This project uses *IMDb movie review* as our primary dataset. The dataset has 50,000 balanced reviews of movies, that is there are 25,000 positive and 25,000 negative movie reviews. The distribution of review length is skewed toward shorter texts (Figure 1 (a)).

We also used *Sentiment140* (Go et al., 2009) and a third self-created *Novels* dataset for testing. The purpose of using these two additional datasets is to evaluate the domain transferability of the machine learning models. The *Sentiment140* dataset has 1,600,000 tweets on Twitter. We extracted 359 tweets to be used as testing set and the remainder was used for training. The *Sentiment140* dataset has three types of labeled sentiments: *positive*, *neutral*, and *negative*. We removed the data labeled as *neutral* for comparison with the other two datasets.

[1] http://www.altavista.com/sites/search/adv

The *Novels* dataset is self-labeled, containing 10 positive and 10 negative experts from *Harry Potter* and *Ulysses*.

The length distributions of *Sentiment140* and *Novels* are shown in Figure 1 (b) and (c). Notably, *Sentiment140* contains texts shorter than the *IMDb movie review*, while the average text length in the *Novels* dataset is closer to that of *IMDb movie review*.



(a) *IMDb Movie Review* Dataset (b) *Sentiment140* Dataset (c) *Novels* Dataset

Figure 1: Shown are the length distributions of datasets Used

We import our data of *IMDb movie review* from Keras (Chollet et al., 2015). Every word is encoded as an integer word index, which is the rank of their frequency in the corpus. A "1" means the most frequent word. When importing, the *vocabulary size* command sets the maximum number of vocabularies imported based on their ranks of frequency. This project sets *vocabulary size = 5,000* which means when importing a review, it only keeps 5,000 most frequent words. Other words are eliminated. This is done because less frequent words contribute less to model's accuracy.

The data processing process proceeds as follows (Figure 2). For a single review, we first convert the text into tokens through a process called tokenization, which takes out individual words, or tokens. The tokenization of a review is performed based on spaces separating individual words. Each token is converted into a string. Some of the tokens will contain punctuation because they are separated by spaces as well and these will be removed. Tokenization and punctuation removal are pre-processed by Keras when importing *IMDb movie review*. However, Python code was written to perform these steps for the two other datasets.

Finally, some of the tokens are so called stop words, such as *and* or *the*. These stop words plays minimal roles in sentiment's representation. Therefore, We remove punctuation and stop words using the Natural Language Toolkit (NLTK) *word_tokenizer* function (Bird et al., 2009).

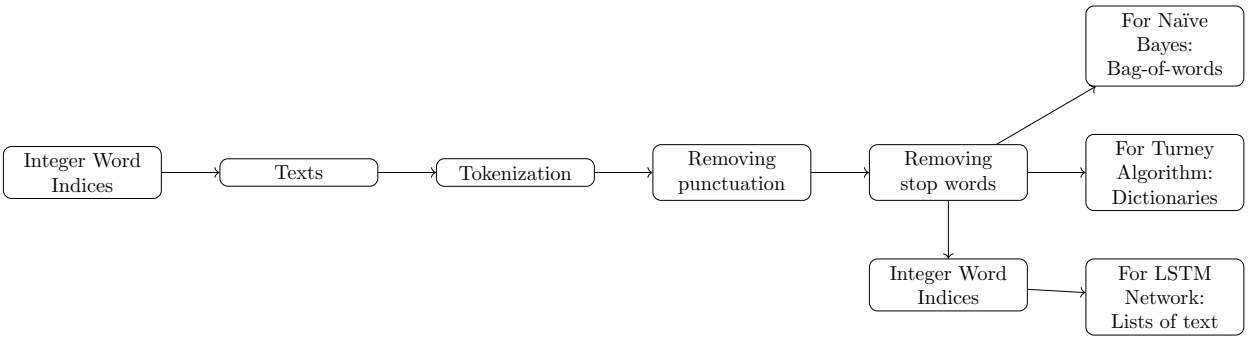


Figure 2: Shown are the steps of data pre-processing to convert unstructured texts into formats suitable for machine learning.

The last step of data pre-processing is to convert tokenized texts with punctuation and stop words removed into the format suitable for each of the three machine learning methods. For Naïve Bayes, the format needs to be in a Bag-of-words representation, in which a review becomes a dictionary with tokenized words being keys, and a boolean value, such as *true*, being dictionary values indicating that the words exist in the review.

For Turney Algorithm, the format needs to be a dictionary with *pos* and *neg* as keys and a 2-D array containing lists of tokenized words in a review as their corresponding values.

Finally, for training of an LSTM network, we need one additional step to convert texts back to encoded word indices due to the requirement of LSTM network's data representation in Keras. The data format is a 2-D array containing lists of word indices representing words in a review. A separate list of a sentiment variable contains corresponding labels for each review.

We now describe the three algorithms used in this research project.

3.2 Naïve Bayes Classifier

Naïve Bayes is a simple probabilistic algorithm based on the Bayes Rule. This algorithm is commonly used as a benchmark algorithm for sentiment analysis.

Naïve Bayes classifies a data point (review) into a class based on its maximum a posteriori (MAP) estimation, which is the class that maximizes $P(c|d)$ according to Equation 2. Essentially, Naïve Bayes goes through all classes and finds the class c_{MAP} that the review is the most likely to be in.

Let w denote individual words in a review d, and V the number of vocabularies, or distinctive words, in the dataset, with Equation 3 and Equation 4, we can compute $P(d|c)$. Here, $count(w_i, c_j)$ is understood as the count of word w_i in class c_j. In Equation 4, we add 1 on the dividend and divisor to make sure that the quotient is not 0. In extreme cases, if there is no word w_i in review d, $\hat{P}(w_i|c_j) = 0$, causing Equation 2 to be 0.

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \tag{1}$$

$$c_{MAP} = argmax_{c\in C} P(c|d) = argmax_{c\in C} \frac{P(d|c)P(c)}{P(d)} = argmax_{c\in C} P(d|c)P(c) \tag{2}$$

$$P(d|c) = \Pi_{w_i \in d} \hat{P}(w_i|c) \tag{3}$$

$$\hat{P}(w_i|c_j) = \frac{count(w_i, c_j) + 1}{\sum_{w\in V}(count(w, c_j) + 1)} = \frac{count(w_i, c_j) + 1}{(\sum_{w\in V} count(w, c_j)) + |V|}, w_i \in d, c_j \in C \tag{4}$$

Denote c to be a class, C to be the set of all possible classes, and d to be a data point. In our case, d represents a single review. The probability of a review being in class c is computed according to Bayes Rule, or Equation 1. We call $P(c)$ the prior distribution of the class, which is our prior knowledge of the class c before we observe the data. We call $P(d|c)$ the likelihood because it represents how likely a review is to be in class c given that we know it is in class c. This is relevant to the number of observations of each word in review d. We call $P(c|d)$ the posterior distribution of the class c because it represents the probability of a class c given that we observed the data d.

3.2.1 A Naïve Bayes Example

Consider for example a dataset of five sentences with stop words removed (Table 1). We need to compute the conditional probabilities of each word in the testing set: *college*, *life*, *so*, *happy*, conditional on positive class and then on negative class, according to Equation 4. Then, for each class that these four words are conditional on, we take the product of $\hat{P}(w_i|c_j)$ for each word w_i, according to Equation 3. We get $P(Positive|Review5) = 0.0000239$, and $P(Negative|Review5) = 0.0000191$, which puts Review 5 in the positive class.

Table 1: Shown are examples of training and testing reviews and their binary labels. The aim is to predict the label for an unseen (testing) review.

	Data	Text	Label
Training	Review 1	"i'm happy"	Positive
	Review 2	"avengers movie so good"	Positive
	Review 3	"hate movie very boring"	Negative
	Review 4	"college so stressful"	Negative
Testing	Review 5	"college life so happy"	?

3.3 Turney Algorithm

The Turney Algorithm is an unsupervised, lexicon-based learning method (Turney, 2002). The first step in Turney Algorithm is to make a part-of-speech tag for each individual word based on Table 2. For example, *"good movie"* will be tagged *"JJ NN"*.

Table 2: Shown are notations of some of the part-of-speech tags used by Turney Algorithm.

Part-of-speech Tags	Meaning
JJ	Adjective
RB, RBR, or RBS	Adverb
NN, or NNS	Noun
VB, VBD, VBN, VBG	Verb

In the second step, we conduct a bi-gram (two-word phrase) extraction according to Table 3. We look at three continuous words in each text, and if we see one of the five patterns from Table 3, we extract the first two words. Thus, *"A beautiful day at Wake Forest"* will become *"beautiful day"* after bi-gram extraction.

Table 3: Shown are part-of-speech bi-grams extracted by Turney Algorithm.

	First Word (Extracted)	Second Word (Extracted)	Third Word (Not Extracted)
1.	JJ	NN, NNS	anything
2.	RB, RBR, RBS	JJ	not NN nor NNS
3.	JJ	JJ	not NN nor NNS
4.	NN, NNS	JJ	not NN nor NNS
5.	RBm RBR, RBS	VB, VBD, VBN, VBG	anything

To implement Turney Algorithm, we do not need a pre-labeled dataset, but we need a set of seed words whose polarities are known. In this research project we use the following positive seed words: *good*, *excellent*, *best*, and *great*, and negative seed words: *bad*, *poor*, *suck*, *terrible*, and *horrible* (Kanna and Pandiaraja, 2019). They are used with the near operator to create a range of texts around seed words and then to find if there are bi-grams occurring within this range (Equation 5).

$$PMI(w1, w2) = log_2(\frac{hits(w1 \text{ NEAR } w2)}{hits(w1)hits(w2)}) \tag{5}$$

Additionally, we compute pointwise mutual information (PMI) to measure how often two words are associated the dataset (Equation 5). With $hits(w)$ being the count of the word w, PMI is calculated based on how often two words are near each other, and how often these two words appear in the dataset. Thus, if a word or phrase has a higher PMI with a positive seed word than with a negative seed word, this word or phrase is more likely too be positive (Equation 6).

$$Polarity(phrase) = PMI(phrase, "good \text{ OR...OR } excellent") - PMI(phrase, "bad \text{ OR...OR } poor") \tag{6}$$

Next, by substituting Equation 5 in Equation 6, we get Equation 7, which is easier to compute and implement. The polarity of a review is computed by averaging the polarity of all bi-gram phrases. If the polarity is positive, then the review is classified as a positive review. Similarly, if the polarity is negative, then the review is classified as a negative review.

$$Polarity(phrase) = log_2(\frac{hits(phrase \text{ NEAR } posSeeds) * hits(negSeeds)}{hits(phrase \text{ NEAR } negSeeds) * hits(posSeeds)}) \\ \text{where } posSeeds = \text{"}good \text{ OR...OR } excellent\text{"} \\ \text{where } negSeeds = \text{"}bad \text{ OR...OR } poor\text{"} \tag{7}$$

3.4 Long Short-term Memory Network

LSTM network is a deep learning method (Hochreiter and Schmidhuber, 1997). It is widely used for text analysis because it captures the information about the order of words in texts. LSTM networks are a type of the Recurrent Neural Network (RNN) (Figure 3[2]), where x denotes inputs (controlled by *input_shape* parameter in Keras), A denotes RNN cells, h denotes hidden units, and t denotes timesteps. An RNN is initially rolled. When data is presented into the RNN, the network will process the data in the unrolled fashion. Batch size is the number of data inputted into the network each time and it contains several movie reviews. For each movie review, word indices will be encoded into $x_0...x_t$ in groups, depending on the *input_shape* parameter. Then, $x_0...x_t$ are passed to the RNN layer, shown in detail in Figure 5[3]. Horizontal arrows illustrate that one RNN unit can pass information to the RNN units in the following timesteps, indicating that RNN is useful for data with long-term dependencies.

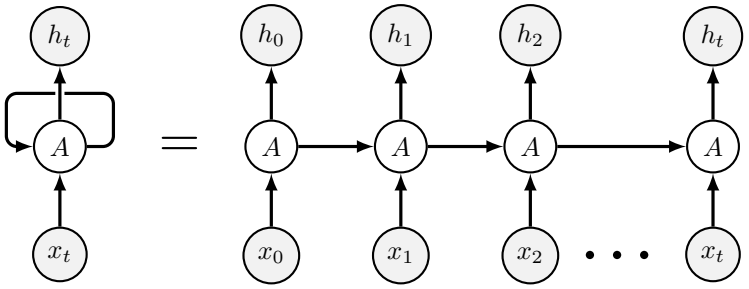


Figure 3: Shown are the rolled and unrolled visualizations of a Recurrent Neural Network.

The architecture of the LSTM network in this project comprises embedding layer, LSTM layer, dropout layer, fully connected layer, and dense layer with sigmoid activation (Figure 4).

Prior to inputting word indices into the LSTM network, the tokenized data is first padded, which sets the review length to be the same for all reviews. If the original review is longer than the required length, we truncated it. If it is shorter, we fill it with empty strings.

Then, in the embedding layer, each word is represented by a numerical vector. Reviews with closer meanings will be embedded closer in vector space because they may have the same numerical values in their vectors.

[2]Figure from: https://tex.stackexchange.com/questions/494139
[3]Figure from https://tex.stackexchange.com/questions/432312

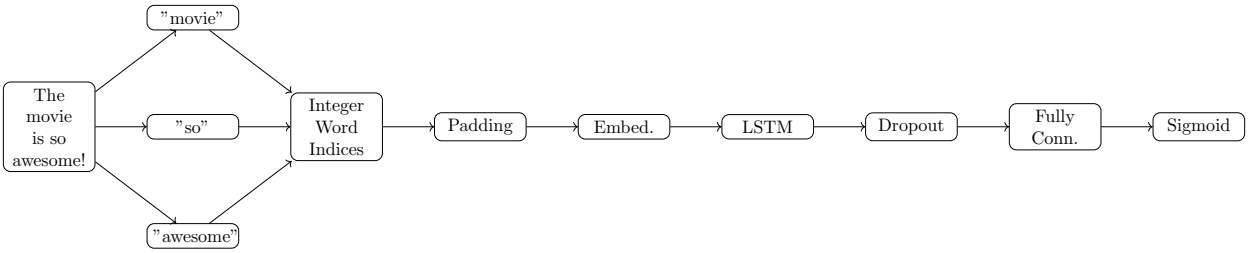


Figure 4: Shown are the inputs to LSTM and model's layers (Embed: Embedding layer; LSTM: Long Short-term Memory layer; Fully Conn.: Fully Connected layer; Sigmoid: Sigmoid activations).

The LSTM layer does the classification (Figure 5). There are 3 sigmoid operations in the LSTM unit. Sigmoid operations control the information flow from cell state, which is information passing through each LSTM unit. Sigmoid operators output a value between 0 and 1, which determines the amount of information of cell state to pass to the next unit, or to discard. Finally, the fully connected layer outputs predictions.

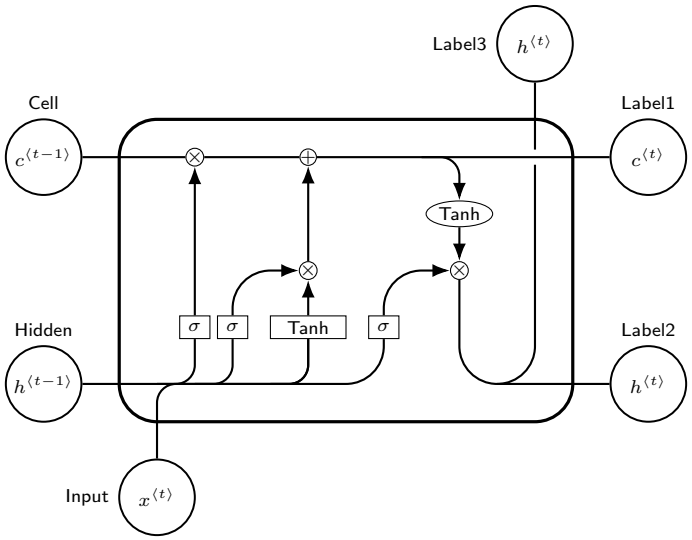


Figure 5: Shown are the elements of an LSTM unit and its input and output.

3.4.1 LSTM Network Parameter Turning

There exist many ways of tuning the parameters in a neural network. In this research project, we tune three parameters in the order of when they are fed into the network. We iterate through various parameter values and select the one resulting in the highest accuracy of sentiment classification (Figure 6). The three parameters and their selected values are: *maximum word length for padding = 160*, *embedding size = 64*, *batch size = 16*.

Besides the three parameters that we tune, other parameters of the LSTM network include: *time step = 2*, which means that the observations used in the previous time step are also used as the input of current time step; *LSTM input unit = 100*, which means we have 100 LSTM neuron units.

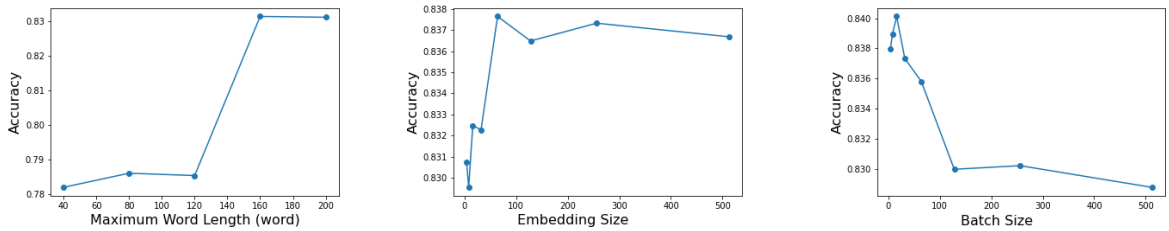


Figure 6: Shown are the plots of accuracy over choices of parameter values. For each parameter, the one resulting the highest accuracy is selected.

3.5 Model Comparison

All models are evaluated using the same sets of metrics (Section 3.6). For each of the three machine learning algorithms, four computational experiments are conducted using four sets of training and testing texts:

1. Training: *IMDb movie review*, Testing: *IMDb movie review* (5-fold cross validation)
2. Training: *IMDb movie review*, Testing: *Sentiment140*.
3. Training: *Sentiment140*, Testing: *Sentiment140*.
4. Training: *IMDb movie review*, Testing: *Novels*.

In experiment 1, we compute the metrics using a 5-fold cross validation method. This experiment evaluates the general performance of the models. We only use cross validation in experiment 1 because *IMDb movie review* is our primary dataset in this analysis. In 5-fold cross validation, the dataset is shuffled and partitioned into 5 subsets or folds. Next, we iteratively use the first fold (20%) of the data for testing and the rest (80%) for training. We repeat this validation process 5 times to ensure that each fold is used as a testing set. We record the performance metrics for each fold, and compute the mean of each metric by averaging across all 5 folds.

In experiment 2, we use the entire *IMDb movie review* dataset with size of 50,000 as training, and use a subset of *Sentiment140* dataset with size of 359 as testing. This experiment evaluates the applicability of the model on a dataset with small text length.

In experiment 3, we use the entire *Sentiment140* dataset with size of 1,600,000 as training, and a subset of *Sentiment140* dataset with size of 359 as testing. This experiment is done for comparison with experiment 2 to evaluate whether type of training data plays a significant role in classifier's accuracy.

In experiment 4, we use the entire *IMDb movie review* dataset with size of 50,000 as training, and a *Novels* dataset with size of 20 as testing. Since the *Novels* dataset has text length distribution similar to *IMDb movie review*, this experiment is done for comparison with experiment 2 to evaluate whether length of testing texts plays a significant role in the results.

3.6 Performance Evaluation

Several performance metrics were computed for each experiment. They were: $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$, $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$, and $F1\text{-}score = \frac{2*Precision*Recal}{Precision+Recall}$, where TP is the number of correctly predicted positive data, TN is the number of correctly predicted negative data, FP is the incorrectly predicted positive data, and FN is the incorrectly predicted negative data. We also record the standard deviation of the accuracy in experiment 1 (5-fold cross validation). Since experiment 1 uses our primary dataset, and accuracy is a good metric, because the dataset is balanced, that is we have the same number of positive and negative texts. We computed the standard deviation for accuracy in experiment 1 to examine the variation in predicted accuracy between the 5 folds.

Note that due to the slow runtime of the Turney Algorithm, in experiment 1, we randomly sample 1,000 instances from the 10,000 testing data in each fold. Additionally, in the computation of polarity in the Turney Algorithm, if we do not observe a seed word or an extracted bi-gram, the equation will return

$PMI = 0$, which makes the data unusable for the classification. In this case, the texts are removed from further evaluation.

We also evaluated whether the LSTM model over-fits the training dataset as follows. We plotted the accuracy and loss as a function of the number of epochs and found that validation accuracy decreases and validation loss increases over epochs (Figure 7). This shows signs of over-fitting because the performance for validation does not exhibit the same pattern. In response to this, we added a dropout layer. As a result, the validation accuracy became flatter, as it did not decrease as fast as previously. However, over-fitting still visible.

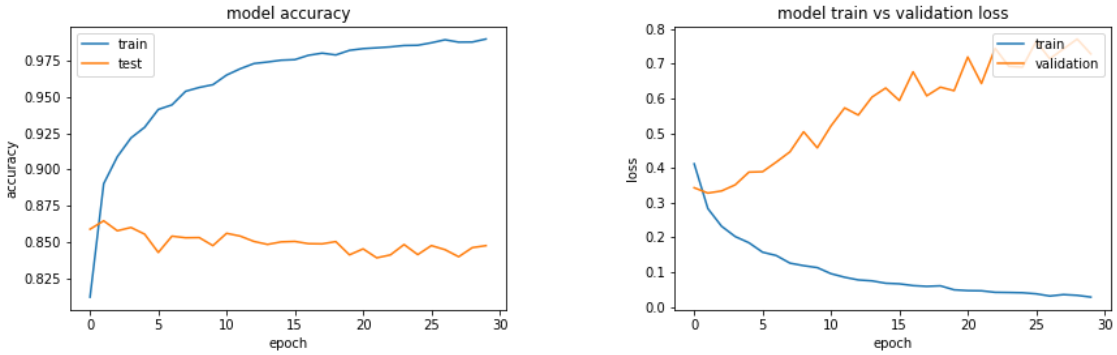


Figure 7: Shown are the plots of accuracy (left) and loss (right) as a function of the number of epochs used for training, after adding a dropout layer.

Next, we discuss the results our experiments.

4 Results

We examined the performance of each model in every experiment. In experiment 1, LSTM network has the highest performance in accuracy (86.17%), precision (87.45%), and F1-score (85.45%), and Naïve Bayes has the highest performance in recall (91.16%) (Table 4). The standard deviations of the accuracy for Naïve Bayes, Turney Algorithm, and LSTM network are 0.0029, 0.011, and 0.016, respectively.

In experiment 2, Naïve Bayes has the highest performance in accuracy (53.48%), recall (60.99%), and F1-score (57.07%). LSTM network has the highest performance in precision (58.07%). This experiment is crucial when we consider domain transferability. We observe that the performance of Naïve Bayes algorithms is the best overall, especially in F1-score. This metric describes the overall performance of both classifying positive sentiments and negative ones. Because it is important to identify both sentiments correctly, this metric provides a better estimate of the strength of each classifier.

In experiment 3, LSTM has the highest performance in accuracy (81.34%) and F1-score (78.86%). Turney Algorithm has the highest performance in recall (81.33%). Naïve Bayes has the highest performance in precision (87.32%). In comparison with experiment 2, we observe in experiment 3 that the performance of all algorithms increased in this experiment, which again indicates an issue for domain transfer.

In experiment 4, LSTM network has the highest performance in accuracy (55.00%) and precision (99.99%). Turney Algorithm has the highest performance in recall (70.00%) and F1-score (53.85%).

5 Discussion

The three models are good representatives of the various types of machine learning methods. Naïve Bayes and Turney Algorithm are shallow learning methods, while LSTM is a deep learning method. The results of Naïve Bayes and Turney Algorithm are easy to interpret due to their modeling processes represented by mathematical equations. LSTM, on the other hand, comprises of several black boxes and its results are to difficult to interpret. Finally, Turney Algorithm is an unsupervised learning method, while Naïve Bayes

Table 4: Shown are the results comparison of the three machine learning methods, trained on three different sets of data, and evaluated on five different metrics.

Method	Dataset Used	Metrics			
		Accuracy	Precision	Recall	F1-score
Naïve Bayes	Training: *movie reviews* Testing: *movie reviews*	83.73%	79.37%	91.16%	84.86%
	Training: *movie reviews* Testing: *Sentiment140*	53.48%	53.62%	60.99%	57.07%
	Training: *Sentiment140* Testing: *Sentiment140*	78.83%	87.32%	68.13%	76.54%
	Training: *movie reviews* Testing: *Novels*	40.00%	42.86%	60.00%	50.00%
Turney Algorithm	Training: *movie reviews* Testing: *movie reviews*	50.87%	50.47%	75.19%	60.39%
	Training: *movie reviews* Testing: *Sentiment140*	37.68%	44.74%	42.50%	43.59%
	Training: *Sentiment140* Testing: *Sentiment140*	65.89%	67.03%	81.33%	73.49%
	Training: *movie reviews* Testing: *Novels*	40.00%	43.75%	70.00%	53.85%
LSTM Network	Training: *movie reviews* Testing: *movie reviews*	86.17%	87.45%	84.69%	85.45%
	Training: *movie reviews* Testing: *Sentiment140*	52.09%	58.07%	40.19%	44.37%
	Training: *Sentiment140* Testing: *Sentiment140*	81.34%	76.16%	82.56%	78.86%
	Training: *movie reviews* Testing: *Novels*	55.00%	99.99%	10.00%	18.18%

and LSTM are supervised learning methods, that is they guided by the availability of labeled sentiments in building of the accurate models.

Evaluating the three methods using accuracy is appropriate because we have a perfectly balanced dataset, with 50% positive reviews and 50% negative reviews. Another important metric is a recall. Recall reflects the performance for the positive class. High recall of Turney Algorithm in this work shows that this method is a good choice for predicting positive sentiments in an unsupervised way. As we observe a fairly large discrepancy between precision and recall in experiment 2 and 4, where training and testing sets are from two different datasets, we are more interested in the performance assessed with an F1-score, which gives us an overall evaluation of how well the model classifies both positive and negative sentiments. LSTM network in experiment 4 exhibits a significantly lower F1-score because of its limitation in classifying positive novels and good performance in classifying negative ones.

The main objective of this research was to determine how well the models learn sentiment from texts unrelated to the application domain. When trained with movie reviews, all three models are unable to successfully recognize sentiments in unrelated contexts. When we compare experiment 2 and 3, which use 2 different training datasets for testing on the same testing set, we observe that the three models are all dependent on what data they use. They can perform well in differentiating movie reviews and tweets, but fails to extract sentiments from them.

Moreover, over-fitting persists in the LSTM network (Figure 7). Because LSTM network learns from long-term dependencies and contextual connections between words, the way the data is phrased is well learned, impacting the performance. Reviewing a movie, sending a Twitter post, and writing a novel all require different wording patterns. Thus, it is hard to use models interchangeably on different datasets.

While Naïve Bayes ignores the position of words, over-fitting also occurs. Mathematically, the calculation of the probabilities in Naïve Bayes depends on whether a word is in the text or not. Similarly, low performance of the Turney Algorithm is also due to its high dependency on whether both a particular bi-gram and a seed word exit in the training data. If the training set simply lacks particular phrases, the accuracy will drop.

The result also shows that Turney Algorithm requires a large corpus for training (Kanna and Pandiaraja, 2019). In its original implementation, the model was trained using the dataset from AltaVista, a search engine with a corpus of a hundred billion words. Therefore, the reported performance of the original Turney Algorithm is very high (Turney and Littman, 2002). In this project, especially in experiment 2, a large number of testing data cannot be classified because of either no bi-grams extracted, or no bi-grams or seed words found in training set. If a collection of a very large training dataset is possible, the Turney Algorithm is advantageous due to its unsupervised nature of learning. However, the long runtime for testing makes it less efficient than Naïve Bayes and LSTM.

Overall, challenges should be expected in using models trained on movie reviews for predicting sentiments with *text2collage* tool due to limited transferrability. The *text2collage* is a method that uses classified emotions from psychiatry texts and then automatically retrieves relevant images to create a collage to help patients visualize their emotions. Among the three algorithms, Naïve Bayes is a preferred method because it has a fairly high accuracy in movie reviews and does not over-fit as much as the LSTM network. It is also fast to train and classify. Moreover, the models are interpretable and may inform better about the classification results.

6 Conclusion

In this work, three machine learning algorithms were evaluated for the task of predicting sentiments in texts. In a 5-fold cross validation experiment with a dataset comprising 50,000 movie reviews, LSTM model had highest accuracy (86%), followed by Naïve Bayes (84%) and Turney Algorithm (51%). The performance of all three models decreased when testing was done on datasets drawn from different domains. Transferability of all models is subject to limitations and hence, it would be challenging to apply one of these methods in the *text2collage* tool. Among the three models, Naïve Bayes performs the best in terms of transferability, with a fairly high F1-score. Therefore, we conclude that Naïve Bayes is the best model for the *text2collage* tool which aims to assist in psychological disease diagnosis and therapies. If the sufficiently large collection of annotated texts could be assembled, LSTM is expected to outperform the other models. For future work, we will improve our LSTM network by focusing on the reduction of its tendency to over-fit the training data.

References

Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly Media, Inc., 1st edition.

Blitzer, J., Dredze, M., and Pereira, F. (2007). Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification.

Chollet, F. et al. (2015). Keras. https://keras.io.

Go, A., Bhayani, R., and Huang, L. (2009). Twitter sentiment classification using distant supervision. *Processing*, 150.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Kanna, P. R. and Pandiaraja, P. (2019). An efficient sentiment analysis approach for product review using turney algorithm. *Procedia Computer Science*, 165:356 – 362. 2nd International Conference on Recent Trends in Advanced Computing ICRTAC -DISRUP - TIV INNOVATION , 2019 November 11-12, 2019.

Kannan, V. and Khuri, N. (2018). Creating digital collages inspired by english texts. In *Proceedings of 13th International Conference on Digital Information Management*, pages 228–233.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Ni, J., Li, J., and McAuley, J. (2019). Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197, Hong Kong, China. Association for Computational Linguistics.

Sachin, S., Tripathi, A., Mahajan, N., Aggarwal, S., and Nagrath, P. (2020). Sentiment analysis using gated recurrent neural networks. *SN Computer Science*, 1(2):74.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Sun, H., Jiang, T., and Dai, Y. (2019). Sentiment analysis of commodity reviews based on multilayer lstm network. In *Proceedings of the International Conference on Artificial Intelligence, Information Processing and Cloud Computing*, AIIPCC '19, New York, NY, USA. Association for Computing Machinery.

Turney, P. D. (2002). Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. *ArXiv*, cs.LG/0212032.

Turney, P. D. and Littman, M. L. (2002). Unsupervised learning of semantic orientation from a hundred-billion-word corpus.