Bi-directional Recurrent Convolutional Neural Network for Opinion Spam Detection

Yitong Chen (CHENY2@Carleton.Edu)
Department of Computer Science, 1 N. College Street
Northfield, MN 55057 USA

Yingqi Ding (DINGY@Carleton.Edu)
Department of Computer Science, 1 N. College Street
Northfield, MN 55057 USA

Abstract

As e-commerce begins to dominate the market, a growing amount of research has been devoted to examine the authenticity of user-generated reviews for online businesses. Machine learning has been successfully used to detect deceptive reviews in experiment settings and therefore has the potential to help purifying the online review environment. Motivated by previous research on automatic review spam detection, we develop a model that combines the bi-directional recurrent neural networks and the convolutional neural networks to classify truthful and deceptive online reviews. We show that our model is capable of effectively distinguishing between spam and non-spam reviews by measuring its performance against novel data that it is not trained on. We compare our model to other popular models that have been used for this purpose including the support vector machine and the convolutional neural network. We find that our model and these alternative models show similar degree of accuracy at identifying review spams. We discuss modifications to the current implementations of the models that should enable further improvements in performance.

Keywords: opinion spam, spam detection, bi-directional recurrent neural network (BRNN), support vector machine (SVM), convolutional neural network (CNN).

Introduction

With the growth of social media and online businesses, people increasingly share their opinions about events and their ratings and reviews of products and services online. However, not all of these reviews represent genuine opinions. In fact, some business owners resort to online reviews to manipulate public opinions: false positive reviews are generated by companies to promote their brands and negative ones to downgrade the reputation of the competitors. Some of these artificially generated reviews are intentionally made to sound authentic, and therefore are hard to be distinguished from the truthful reviews. They can easily misguide users into making unintended shopping decisions.

These misleading and unhelpful online reviews are referred to as *review spam* or *opinion spam* by Jindal and Bing (2008). Jindal and Bing (2008) identifies three types of spam reviews for merchandise in online shopping platforms: untruthful opinions, reviews on brands only and non-reviews. This paper focuses on the first type of review spam. In particular, humans are found to perform almost at chance when asked to classify truthful and fraudulent reviews (Ott, Choi, Cardie and Hancock 2011). Thus, it is especially important to develop a mechanism that helps to detect such spam information and purify the online review environment.

Many machine learning algorithms have been developed to solve this problem and the broader problem of text classification. The first challenge that researchers faced is how to represent the meaning of a given piece of text. In general, two approaches are used. The first approach represents a piece of text as *strings*, i.e. a sequence of words. The second is the bag-of-words (BoW) approach, which, as the name suggests, represents a piece of text (e.g. a sentence or a document) as a bag or multiset of all its words, ignoring the grammar and the order of words but only keeping the multiplicity (*Wikipedia*, Bag-of-words model). The BoW approach is commonly used in machine learning because of its simplicity, but the simple form of BoW is not sufficient since the meaning of a word crucially depends on the context that it appears in and contextual information is ignored by BoW. To take an example from Lai, Xu, Liu and Zhao (2015), given the sentence *"A sunset stroll along the South Bank affords an array of stunning vantage points"*, the uni-gram "*Bank*" is ambiguous as it can refer to either a financial institution or the land by a body of water. A human speaker would potentially need the whole phrase "*A sunset stroll along the South Bank*" to completely disambiguate the meaning of "*Bank*", and therefore, a successful model for text representation will need to be able to encode the contextual information as well. Accurate semantic representation is the prerequisite for correctly classifying the text input.

Both the Support Vector Machine (SVM) Classifier and the Neural Network (NN) Classifier are capable of encoding semantic representations and both have used for opinion spam detection. Both of these two models are discriminative models (Aggarwal and Zhai 2012). The SVM Classifiers aim to partition the search space by finding class separators that best simulate the boundaries between different classes (Cortes and Vapnik 1995). As demonstrated by Ott et al. (2011) and Xu and Zhao (2012), SVM classifiers are well-suited for text classification tasks such as opinion spam detection since they can handle the high-dimensional and sparse features typical of text inputs.

The Neural Network Classifiers can also effectively approximate the class boundary functions but do so through error-based weight adjustments in the network. Specifically for text processing, the Recurrent Neural Network (RNN) proposed by Elman (1990) is capable of encoding temporal information such as the context that a word appears in. The Elman Nets, however, have the problem of biasing toward the words that appear later than earlier (Lai et al. 2015). To overcome this problem, we follow Lai et al. (2015) and Ren and Ji (2017) and combine the Recurrent Neural Network with Convolutional Neural Network (CNN) since the latter helps eliminating bias through a max-pooling layer. Instead of using the traditional uni-directional RNN, we use a bi-directional one as it captures both the left and the right context when encoding semantic information. A high level representation of our network can be seen in Figure 1.

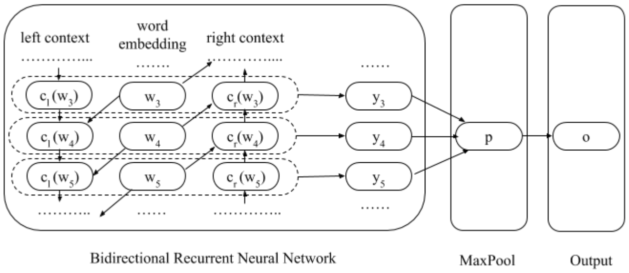


Figure 1: The structure of the recurrent convolutional neural network. Adapted from Lai et al. (2015).

We compare the performance of our RCNN model with a model that uses only a CNN and a SVM Classifier model in detecting opinion spams. We train and test each model using labeled review data collected by Ott et al. (2011). The results show that neural networks and SVMs are comparable in terms of their abilities to correctly classify spam reviews. In terms of efficiency, the neural network models turn out to be relatively slower to train than the SVM models. Therefore, in the cases where both training data and training time are limited, the SVM might be the better classifier to use for opinion spam detection.

Models

Neural Networks

Word Embeddings The first step of building a neural network model that processes text input is transforming words in the input text into vectors. The most straight-forward approach is to use one-hot encoding. There are two problems with this approach. First, the dimension of the one-hot encoded vectors will be fairly large, effectively the size of the vocabulary. The resulting vectors will be very sparse as each vector will contain a 1 at one index and 0s at all other indices. The more severe problem with one-hot encoding is that it is not able to capture all the information available in the input. For instance, the one-hot encoded vectors for ”dog” and ”cat” will be completely unrelated to each other despite that the two words frequently appear in the same context.

Instead of one-hot encoding, we choose to build a Word2Vec model (Mikolov, Sutskever, Chen, Corrado and Dean 2013) to obtain smaller and denser embedding vectors of words that contains richer information. Specifically, we use the Skip-Gram version of Word2Vec described by Mikolov et al. (2013). With the Skip-Gram model, the neural network is trained to predict a window of neighboring words of the current word. For example, if the data is the sentence ”*I went to the store yesterday*”, a Skip-Gram model with window size 1 will learn to predict the words ”to” and ”store” given the current word ”the”. With more data than a single sentence, the model will predict the words with highest probability out of a set of valid neighbors. Prediction mistakes are used to train the model through back propagation. Specifically, for words $w_1, ..., w_n$ in the vocabulary, the model aims to maximize the average log probability

$$\frac{1}{n}\Sigma_{i=1}^{n}\Sigma_{-c\leq j\leq c, j!=0}\log p(w_{i+j}|w_i)$$

where c is the window size.

Bi-directional RCNN We implement a bi-directional recurrent convolutional neural network for the actual classification task. The task is broken down into several sub-problems that different parts of the network aim to solve. The bi-directional recurrent structure is responsible for generating the semantic representation of a word. It does so by combining the vector embedding of the word w_i and the encoding of its left and right contexts $c_l(w_i)$ and $c_r(w_i)$, as shown in Figure 1. Bi-direction RNNs are able to encode more context information than uni-directional RNNs, and the additional information may be used to disambiguate between different meanings of a given word. A classic pair of sentences that have been used to demonstrate the power of bi-directional RNNs are “*He said, Teddy bears are on sale*” and “*He said, Teddy Roosevelt was a great President*”. In this example, if the model only encodes the preceding context, it would not be able to distinguish between “*Teddy*” as in “*Teddy bears*” and “*Teddy*” as in “*Teddy Roosevelt*”. Bi-directional RNN allows the model to look ahead and incorporate the sequences following the current word into its underlying representation of words.

Instead of using simple RNN cells for the context layers of the bi-directional RNN, we use Long short-term memory cell (LSTM) (Gers, Schmidhuber and Cummins 2000). The advantage of using LSTM is that it allows for encoding long distance dependencies, which are common in text data.

The output of the bi-directional recurrent layers, i.e. the semantic representation of words, are fed into a pooling layer. The pooling layer is responsible for selecting the significant semantic factors in a piece of review. We adopt the reasoning from Lai et al. (2015) for using a max pooling in this layer instead of the alternatives such as average pooling. Intuitively, as human readers, it is possible for us to get the semantic core of a piece of text by just reading a few key words. The job of the max pooling layer in our model is essentially to pick out these key words for the network so that it can construct a semantic representation of the whole piece of text.

Finally, the output of the max pooling layer will be fed into the final, fully-connected output layer. The output layer contains two nodes o_0 and o_1. The activation on o_0 represents the relative likelihood that the current piece of text is a deceptive review, while the activation on o_1 represents the relative likelihood for the piece of text to be a truthful review. The predicted label is the the one that maps to the cell with a higher activation value.

Convolutional Neural Network (CNN) Convolutional neural networks without the recurrent layers have been used to detect opinion spams by Li, Qin, Ren and Liu (2016). We take their implementation of the model and compare with our bi-directional RCNN. Li et al. (2016) refers to their model as the sentence convolutional neural network (SCNN), and the high-level structure of the network can be seen in Figure 2.

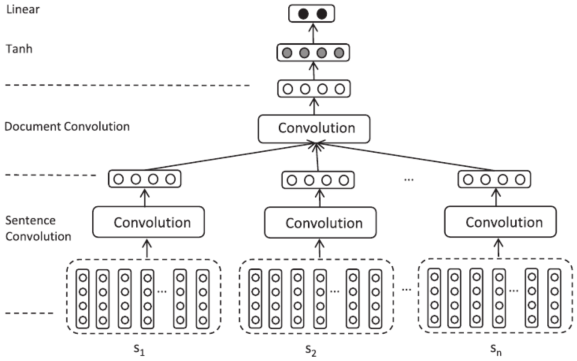


Figure 2: SCNN model for learning sentence representation (Li et al. 2017)

SCNN contains two convolutional layers, one for sentence convolution and the other for document convolution. The first layer makes a composition of each sentence using a fix-length window, while the second layer transforms vectors of the sentences into a vector of document. For example, given the input sentence *"The Chicago Hilton is great"*, the model will first map the words into corresponding word embeddings, just like our RCNN model. Then the convolutional layer extracts local features based on the words' semantic meaning. The first pooling layer combines these local feature vectors to obtain a global feature vector. The second pooling layer combines the sentence vectors into a single document through a weighted-average pooling operation. The penultimate nonlinear layer captures high level features, while the final linear layer is designed to compute the scores for different categories.

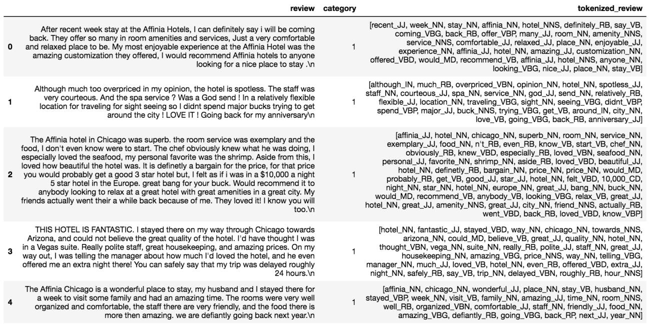
Support Vector Machine

Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outlier detection. They are capable of doing a non-linear classification efficiently with the kernel trick, which maps the inputs into high-dimensional feature spaces. This property allows it to perform fairly well at text classification. Ott et al. (2011), for instance, uses SVM to build an opinion spam detector with more than 90% accuracy.

The SVM we use in this project is designed based on a number of general POS features that can shed light on a general rule for identifying deceptive opinions. The POS features are proposed based on findings (Li et al. 2017) such as truth reviews usually contain more nouns (N), adjectives (JJ), prepositions (IN) and determiners (DT), while spam reviews tend to contain more verbs (V), adverbs (RB), pronouns (PRP) and pre-determiners (PDT). Raw data is tagged by an automatic parser whose output can be seen in Figure 3. The tagged training data is fed into the model. The algorithm outputs an optimal hyper-plane that categorizes new examples, which in 2D space will be a line that divides the plane into two parts where each class lies in either side. In our project, the labels include information about whether the opinion is positive or negative, truthful or deceptive, giving rise to four classes as shown in Table 1. The reviews are classified based on how extreme they are, either positively or negatively. In other words, if the opinion belongs to the first or forth category in Table 1, then it is classified as deceptive; if the opinion belongs to the second or third category, then it is classified as truthful.

Category	Label	#
Highly Positive	Positive, Deceptive	400
Positive	Positive, Truthful	400
Negative	Negative, Truthful	400
Highly Negative	Negative, Deceptive	400

Table 1: Categories of Reviews



	review	category	tokenized_review
0	After recent week stay at the Affinia Hotels, I can definitely say i will be coming back. They offer so many in room amenities and services, Just a very comfortable and relaxed place to be. My most enjoyable experience at the Affinia Hotel was the amazing customization they offered, I would recommend Affinia hotels to anyone looking for a nice place to stay .\n	1	[recent_JJ, week_NN, stay_NN, affinia_NN, hotel_NNS, definitely_RB, say_VB, coming_VBG, back_RB, offer_VBP, many_JJ, room_NN, amenity_NNS, service_NNS, comfortable_JJ, relaxed_JJ, place_NN, enjoyable_JJ, experience_NN, affinia_JJ, hotel_NN, amazing_JJ, customization_NN, offered_VBD, would_MD, recommend_VB, affinia_JJ, hotel_NNS, anyone_NN, looking_VBG, nice_JJ, place_NN, stay_VB]
1	Although much too overpriced in my opinion, the hotel is spotless. The staff was very courteous. And the spa service ? Was a God send ! In a relatively flexible location for traveling for sight seeing so I didnt spend major bucks trying to get around the city ! LOVE IT ! Going back for my anniversary\n	1	[although_IN, much_RB, overpriced_VBN, opinion_NN, hotel_NN, spotless_JJ, staff_NN, courteous_JJ, spa_NN, service_NN, god_JJ, send_NN, relatively_RB, flexible_JJ, location_NN, traveling_VBG, sight_NN, seeing_VBG, didnt_VBP, spend_VBP, major_JJ, buck_NNS, trying_VBG, get_VB, around_IN, city_NN, love_VB, going_VBG, back_RB, anniversary_JJ]
2	The Affinia hotel in Chicago was superb. the room service was exemplary and the food, I don't even know were to start. The chef obviously knew what he was doing, I especially loved the seafood, my personal favorite was the shrimp. Aside from this, I loved how beautiful the hotel was. It is definetly a bargain for the price, for that price you would probably get a good 3 star hotel but, I felt as if i was in a $10,000 a night 5 star hotel in the Europe. great bang for your buck. Would recommend it to anybody looking to relax at a great hotel with great amenities in a great city. My friends actually went their a while back because of me. They loved it! I know you will too.\n	1	[affinia_JJ, hotel_NN, chicago_NN, superb_NN, room_NN, service_NN, exemplary_JJ, food_NN, n't_RB, even_RB, know_VB, start_VB, chef_NN, obviously_RB, knew_VBD, especially_RB, loved_VBN, seafood_NN, personal_JJ, favorite_NN, shrimp_NN, aside_RB, loved_VBD, beautiful_JJ, hotel_NN, definetly_RB, bargain_NN, price_NN, price_NN, would_MD, probably_RB, get_VB, good_JJ, star_JJ, hotel_NN, felt_VBD, 10,000_CD, night_NN, star_NN, hotel_NN, europe_NN, great_JJ, bang_NN, buck_NN, would_MD, recommend_VB, anybody_VB, looking_VBG, relax_VB, great_JJ, hotel_NN, great_JJ, amenity_NNS, great_JJ, city_NN, friend_NNS, actually_RB, went_VBD, back_RB, loved_VBD, know_VBP]
3	THIS HOTEL IS FANTASTIC. I stayed there on my way through Chicago towards Arizona, and could not believe the great quality of the hotel. I'd have thought I was in a Vegas suite. Really polite staff, great housekeeping, and amazing prices. On my way out, I was telling the manager about how much I'd loved the hotel, and he even offered me an extra night there! You can safely say that my trip was delayed roughly 24 hours.\n	1	[hotel_NN, fantastic_JJ, stayed_VBD, way_NN, chicago_NN, towards_NNS, arizona_NN, could_MD, believe_VB, great_JJ, quality_NN, hotel_NN, thought_VBN, vega_NN, suite_NN, really_RB, polite_JJ, staff_NN, great_JJ, housekeeping_NN, amazing_VBG, price_NNS, way_NN, telling_VBG, manager_NN, much_JJ, loved_VB, hotel_NN, even_RB, offered_VBD, extra_JJ, night_NN, safely_RB, say_VB, trip_NN, delayed_VBN, roughly_RB, hour_NNS]
4	The Affinia Chicago is a wonderful place to stay, my husband and I stayed there for a week to visit some family and had an amazing time. The rooms were very well organized and comfortable, the staff there are very friendly, and the food there is more then amazing. we are defiantly going back next year.\n	1	[affinia_NN, chicago_NN, wonderful_JJ, place_NN, stay_VB, husband_NN, stayed_VBP, week_NN, visit_VB, family_NN, amazing_JJ, time_NN, room_NNS, well_RB, organized_VBN, comfortable_JJ, staff_NN, friendly_JJ, food_NN, amazing_VBG, defiantly_RB, going_VBG, back_RP, next_JJ, year_NN]

Figure 3: Tokenized Data Example

Simulation

Dataset

We use the *Deceptive Opinion Spam Corpus v*1.4 released in Ott et al. (2011) as raw inputs for both model training and testing. This corpus is the first large-scale, publicly available data set for deceptive opinion spam research. It contains 400 truthful positive reviews from TripAdvisor, 400 deceptive positive reviews collected through Mechanical Turk, 400 truthful negative reviews from Expedia, Hotels.com, Orbitz, Priceline, TripAdvisor and Yelp, and 400 deceptive negative reviews collected through Mechanical Turk.

Training the Models

Word2Vec The Word2Vec model is built by adapting the basic_word2vec model from TensorFlow. To train the Word2Vec model, a dictionary that contains all possible words from the 1600 pieces of reviews are generated. The dictionary maps each word to a distinctive index. Sequences of words from the reviews, represented with the indices, are fed into the network in order. Using the domain-specific texts as opposed to a general-purpose corpus as the training data means that the word embeddings generated by the model will be specialized for this task.

RCNN The bi-directional RCNN classifier is built using PyTorch, and is adapted from an implementation by Prakash Pandey. Multiple parameters can be set when training the bi-directional RCNN classifier, including the learning rate η, the batch size and the size of the hidden layers. We explore using different batch size in training, which will be discussed in detail in the next sections. In each epoch, tensors that represent batches of reviews, each of which is a sequence of word embeddings generated by the pre-trained Word2Vec are fed into the model. The predicted label is compared with the actual label of the review. The cross entropy loss function

$$-y(\log(p) + (1-y)\log(1-p))$$

is used for computing the loss and back-propagation.

SCNN The SCNN implementation we take from Li et al. (2017) contains its own implementation of word-embedding generators. The reviews, represented as sequences of words are fed into the model. *Tanh* is used as the activation function for the network, and the output of the network is a series of scores for each category. A hinge loss instead of a softmax function as it has more a relaxed constraint. The original loss function used by Li et al. (2017) is

$$Loss(r) = max(0, m_\delta - f(r_t) + f(r_{t^*}))$$

where t is the golden label of the review r, t^* stands for a different non-golden standard label, and m_δ is the margin in the experiment. In our data set, all labels are treated as golden standard labels and thus the term $f(r_{t^*})$ is automatically set to 0.

SVM The SVM model is implemented using libraries including Pandas (data framework), NLTK (stopword removal, tokenizing, tagging, lematization), Gensim (bag of words, corpus-to-sparse convertion), and scikit-learn (GridSearchCV, Random Forest, SVM). When training the SVM, an automatic parser is first used to generate the tagged data. Four-fold cross validation is used in the training to avoid over fitting problem during the supervised learning. Cross-validation is done by giving the model a set of labeled data for training and a set of unknown data to test against. This allows the model to statistically assess how its prediction generalizes to an independent data set and avoid selection bias. We use the GridSearchCV method to automatically search through the parameter space and select the most appropriate parameters to adjust. The Support Vector Classifier (SVC), parameter grids and specified number of cross validations are passed to the GridSearchCV method.

Procedure

For the bi-directional RCNN model, we run a total of three training sessions with batch size 16, two for 10 epochs and one for 20 epochs. We also run a total of three training sessions with batch size 32, two for 10 epochs and one for 30 epochs.

For the SCNN model, we run one training session with 2000 steps. We run one training session for SVM as well.

Results

The average performance of bi-directional RCNN model, trained with batch size 16 and 32 respectively for 10 epochs are shown in Figure 4. The performance of the RCNN model trained with batch size 32 for 30 epochs is shown in Figure 5. We observe an overall increasing trend in accuracy over the course of the training.

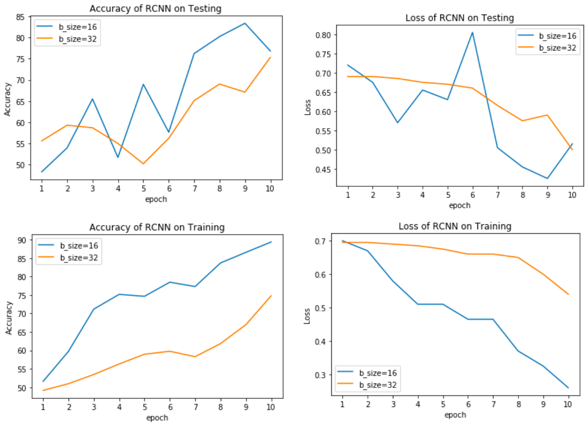

Accuracy of RCNN on Testing
Loss of RCNN on Testing
Accuracy of RCNN on Training
Loss of RCNN on Training

Figure 4: loss and accuracy of the training and testing set for the bi-directional RCNN over 10 epochs

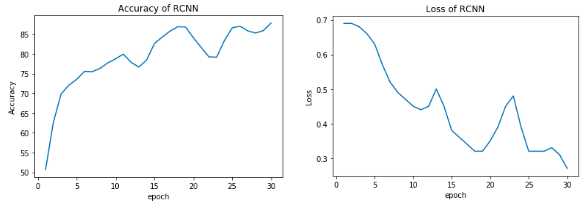

Accuracy of RCNN
Loss of RCNN

Figure 5: loss and accuracy of the training set for the bidirectional RCNN over 30 epochs

The performance of the SCNN model over the course of 2000 training steps is shown in Figure 6. The accuracy of

	Training Data	Testing Data
RCNN	92.43%	85.2%
SCNN	100%	83.5%
SVM	94.3%	77.8%

Table 2: Maximum Idenfication Accuracy for RCNN, SCNN and SVM

the SCNN model reaches its maximum around 400 steps into the simulation. The loss of the SCNN model also converges around 500 steps into the simulation.

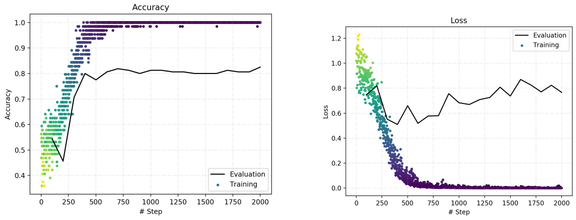


Figure 6: loss and accuracy of SCNN on the training set for 2000 steps and the evaluating set for every 100 steps

The best accuracy for identifying opinion spams in the training set and the testing set for RCNN, SCNN and SVM are shown in Table 2.

Analysis

Final Performance As can be seen from Table 2, the bi-directional RCNN model slightly outperforms the SCNN model, which in turn slightly outperforms the SVM model in correctly identifying opinion spams and non-spams in the testing data. The difference in performance between models are nonetheless not very significant. Interestingly, when classifying the training data, the SCNN model performs the best and is able to reach 100% accuracy, which never happen with the other two models.

As can be seen from Figure 4, the two versions of the RCNN model, one trained with batch size 16 and the other trained with batch size 32, show an interesting comparison in their final performance. The model trained with smaller batch size performs significantly better at classifying deceptive and truthful reviews in the training set, but the two models do not differ from each other in classifying the testing set. In addition, while the RCNN trained with batch size 16 is always better at categorizing the training set than the testing set, the RCNN model trained with batch size 32 shows higher accuracy for the testing set.

Learning Curve For the bi-directional RCNN model, there is a clear contrast between how its performance changes over time for the testing set (upper Figure 4) and for the training set (lower Figure 4). With the training set, the growths of the accuracy of the bidirectional RCNN with b_size = 16 trained for 10 epochs and the bidirectional RCNN with b_size = 32 trained for 30 epochs roughly approximate a logarithmic function, with bigger improvements in earlier epochs and smaller ones approaching the end of the session. The decrease in loss shows a corresponding pattern. With the testing data, it seems that while the accuracy of classification increases overall, the increase is inconsistent and accuracy may drop randomly between epochs.

For the SCNN, it can be clearly seen that the accuracy of the model increases logarithmically and the loss decreases exponentially with the training set. With the testing/evaluation set, the accuracy shows a curious drop in the first 200 steps and then starts increasing logarithmically. There does not seem to be a pattern in the change of the loss over time for the testing set.

Classification Errors To see whether there exists any pattern in the mistakes that the bi-directional RCNN model are making, we run the RCNN model trained with batch size 16 for 20 epochs through the entire data set and examine the reviews that the model misclassifies. A distribution of the mistakes over the two categories is shown in Figure 7.

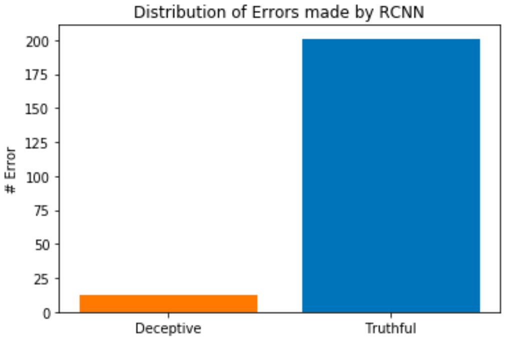


Figure 7: Distribution of RCNN Errors.

Curiously, the amount of truthful reviews that are misclassified as spams greatly exceeds that of deceptive reviews misclassified as genuine opinions. It seems that the RCNN model might be biased toward identifying reviews as opinion spams.

Another characteristics that seems to be shared by the misclassified reviews is that most of these reviews are relatively short and are thus massively padded with ”UNK” values at the end to match the length of the longest review in the same batch when they are fed into the model for training. It is possible that there is just not enough information in the shorter reviews for the network to pool out and make a correct judgment.

Discussion

Comparative Study

Human vs. Model As previously mentioned, opinion spam detection differs from other text classification tasks such as speech sentiment detection in which humans are found to perform poorly at classifying spam and truthful opinions (Ott et al. 2011). Our simulation results reveal that all the machine learning models investigated in this paper perform much better than humans. This suggests that there exist some latent patterns within the deceptive and authentic reviews that are captured by the machine learning models but are hard for humans to discover. Further research is required to understand what these latent patterns are.

Humans and bi-directional RCNN models also differ with respect to the classification errors they make. Ott et al. (2011) found that humans show a strong truth bias in their judgments. In fact, one of the human subjects in their experiment only identify fewer than 12% of the testing items as deceptive even though there are equal number of truthful and deceptive reviews in the testing set. The bi-directional RCNN model is the opposite of humans as it shows a strong tendency to misidentify truthful opinions as deceptive (Figure 7). In practical applications, this bias shown by the RCNN model could be a merit. For instance, if machine learning models are used to identify and delete the fraudulent reviews from online shopping platforms to avoid misguiding the customers, the RCNN model might be preferred over a model that fails to remove many of the deceptive reviews. However, if the models are used to identify and penalize users who frequently generate opinion spams, then using the RCNN model might greatly harm user experience.

Machine Learning Models The results of our simulation suggest that both the neural network model and the SVM are capable of detecting opinion spams with relatively high accuracy. This is consistent with what have been reported by previous research including Ott et al. (2011) and Ren and Ji (2017).

The competitive performance of the SVM model is expected as they are in general very efficient in classifying categories in high dimensional space. A significant advantage of the SVM classifiers is that they are easy to implement and efficient in both space and run time. The state-of-the-art SVM implementations, such as the one we used in our simulations, allow users to add customized kernels, making the models even better suited for the specific categorizing tasks that they are trained for.

One potential problem with the SVM classifier is that calculating the cross-validations is part of its training process. In our simulation, we have 1,600 pieces of data, and calculating cross-validation does not significantly slow down the training process, but with an increased amount of input data, this calculation can become very costly. The controversy here is that by the curse of dimensionality (Bellman 1961), which indicates that the number of samples needed to estimate a function at a given accuracy grows exponentially with the dimensionality of the function (Chen 2009), the SVM model will not be able to maintain its high accuracy given more complicated categorizing tasks unless it is trained with more data. Another problem with the SVM classifier is that it requires tagged input data or at least an automatic parser that can be used to tag raw data. While automatic parsers have been developed for popular languages such as Mandarin and English, many of the historically understudied minority languages cannot be automatically tagged yet, raising equity concerns with the SVM models.

Between the two neural network models, we originally hypothesized that the bi-directional RCNN classifier would significantly outperform the CNN classifiers since we expected the former to be better at encoding semantic information. This hypothesis is not supported by the results of the simulation. This is potentially because the SCNN model we adopt from Li et al. (2017) is specifically designed to overcome the limitations of traditional CNNs in processing text input. Traditional CNNs typically use a fix-length window as the convolutional kernel (Collobert et al. 2011, among others). The problem with this approach is that it is hard to determine the optimal window size such that the network is reasonably fast to train and can capture just the right amount of semantic information. By separating the sentence convolution and the document convolution into two layers, the SCNN model is able to use larger windows without introducing a huge increase in the amount of computations required. Convolutional layers with long windows as kernels are as effective as the bi-directional recurrent layers in encoding contextual information, making up the disadvantage of the SCNN model.

Limitations and Future Research

In this section, we discuss a few limitations of this project and suggest how we can overcome these limitations in future research.

Improving the Training Data The data set used in this project comes from a single domain, i.e. reviews for hotels. Previous research such as Ren and Ji (2017) studies opinion spams in multiple domains including hotel, restaurant and doctor etc.. It will be meaningful to conduct both in-domain and cross-domain comparisons between the three different machine learning models if we can get access to more diversed review data.

Improving the Research Methodology In the previous section, we indicate that the differences between the performance of the machine learning models are less significant than we had anticipated. This result would be more convincing had we controlled the amount of training that each model received.

While the RCNN model shows relatively good performance after training for 30 epochs, the accuracy and the loss value has not converged yet. Unfortunately, we were not able to run more than 30 epochs limited by the memory of our laptops. We expect that training the RCNN network for more epochs using cloud services might improve its performance even further.

As shown in Figure 4, the RCNN model shows fairly random patterns in terms of how its performance on classifying the testing data changes over the 10 epochs. Figure 4 shows the average accuracy and loss value of two independent trials. Running more trials and taking the average over these trials might help eliminating the randomness and allow us to see a clearer pattern regarding how the RCNN model develops over time.

Improving the Models The bi-directional RCNN model we implemented shows compatible performance with the neural network model constructed by Ren and Ji (2017). The accuracy of the SVM classifier created by Ott et al. (2011), on the other hand, is around 95%, much higher than than our SVM model with an accuracy of around 80%. We believe that the features included in the models might contribute to this performance difference. Both Ott et al.'s (2011) model and our model use POS as a feature for classification. But Ott et al. (2011) additionally extracts features such as Linguistic Inquiry and Word Count (LIWC). This suggests that one possible way to improve our model will be to add more features that we can extract from the input texts to the SVM.

Future Works We believe that it will be fairly easy to adapt each of the models we developed for opinion spam detection to other text classification tasks such as detecting the polarity of online reviews. We expect the main difference will be that a new set of labels will be used, but the nature of the classification remains the same.

References

Aggarwal, C. C., & Zhai, C. (2012). A survey of text classification algorithms. In C. C. Aggarwal & C. Zhai (Eds.), *Mining text data* (pp. 163–222). Springer Science+Business Media. doi: 10.1007/978-1-4614-3223-4_6

Bellman, R. (1961). *Adaptive control processes*. Princeton, NJ: Princeton University Press.

Chen, L. (2009). Curse of dimensionality. In L. Liu & M.T.zsu (Eds.), *Encyclopedia of database systems.* Boston, MA: Springer.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & gmai P. Kuksa. (2011). Natural language processing (almost) from scratch. *JMLR*, *12*, 24932537.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*, 273-297.

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with lstm". neural computation. *Neural Computation*, *12 (10)*, 24512471. doi: 10.1162/089976600300015015

Jindal, N., & Liu, B. (2008). Opinion spam and analysis. In *Proceedings of the international conference on web search and web data mining - wsdm 08.* doi: 10.1145/1341531.1341560

Lai, S., Xu, L., Liu, K., & Zhao, J. (2015). Recurrent convolutional neural networks for text classification. In *Proceedings of the twenty-ninth aaai conference on artificial intelligence* (pp. 2267–2273).

Li, J., Ott, M., Cardie, C., & Hovy, E. (2014). Towards a general rule for identifying deceptive opinion spam. *ACL.*

Li, L., Qin, B., Ren, W., & Liu, T. (2017). Representation and feature combination for deceptive spam review detection. *ScienceDirect Neurocomputing journal*, 33-41.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *NIPS*, 3111–3119.

Ott, M., Choi, Y., Cardie, C., & Hancock, J. T. (2011). Finding deceptive opinion spam by any stretch of the imagination. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*, 309319.

Ren, Y., & Ji, D. (2017). Neural networks for deceptive opinion spam detection: An empirical study. *Information Sciences*, *385-386*, 213-224. doi: 10.1016/j.ins.2017.01.015

Wikipedia. (n.d.-a). *Bag of words model.* `https://en.wikipedia.org/wiki/Bag-of-words_model`.

Wikipedia. (n.d.-b). *Part of speech tagging.* `https://en.wikipedia.org/wiki/Part-of-speech_tagging`.

Xu, Q., & Zhao, H. (2012). Using deep linguistic features for finding deceptive opinion spam. *COLING.*