**Sentiment Analysis using Deep Learning techniques**

by Julia Rose Thee, Kuohsun Tsai, Barry L Armour, Ye Chen, and Adelard Banza

CIS600 Fundamentals of Data Mining – Spring 2019

Syracuse University

1. **Introduction**

Sentiment analysis is a classification technique within data mining comprised of many computer science disciplines. Also known as *Opinion Mining*, sentiment analysis combines the fields of natural language processing, machine learning, and text analysis to extract and analyze subjective opinions within textual data. As the primary goal of natural language processing is designing and programming artificially intelligent machines to analyze and understand natural language, sentiment analysis has become a popular application of this field. Although the discipline of natural language processing is concerned with programming computers to understand all facets of natural language (including generating, processing, and interpreting speech, text, emotions, etc.), sentiment analysis is primarily concerned with analyzing, processing, and classifying information within text. More specifically, sentiment analysis works to classify personal opinions and sentiment contained within textual data.

The amount of textual information available over the internet is growing at a rapid pace every day. Within this vast amount of public information on the Internet, there seems to be a limitless number of written texts containing the opinions and sentiment of countless individuals. Subjective, personal opinions are constantly being shared over the internet via platforms such as social media, forums, survey responses, review sites, and blogs. The opinions shared have an endless range of topics—movie reviews, healthcare recommendations, food preferences, and so many others. The primary objective of sentiment analysis is extracting and analyzing the opinions and sentiment contained within these texts on the Internet. The ability of sentiment analysis to distinguish the tone of the sentiment being expressed in text, whether it be a positive, negative, or neutral tone, allows this technique to be used in a wide range of commercial applications. Sentiment analysis can be applied to improving marketing strategies, customer service, product distribution, and numerous other business applications.

One particular application of sentiment analysis, in which this technique has been very successful, is the implementation of recommendation systems. These types of systems are widely utilized by companies such as Amazon, YouTube, Facebook, Spotify, and several others to provide recommendation services (recommending products, music, videos, content, etc.) to users based on their preferences. As a more specific example, Amazon Prime and Video may utilize a user’s search history, purchase history, product ratings or movie reviews to determine the product and movie preferences of the user. After determining a user’s preferences, Amazon would be able to provide useful product and movie recommendations to the user, which may then result in the user making future Amazon purchases. The example just described depends on the user preferences being accurately predicted, which is where a technique such as sentiment analysis comes into play. Sentiment analysis would focus on analyzing the sentiment contained in a user’s product and movie reviews to accurately predict their preferences and make useful future recommendations to the user.

The primary focus of this data mining project would be very useful to a company such as Amazon Video, YouTube, or any other organization centered around films, television, or videos. This project explored using sentiment analysis on the textual data contained in IMDb (Internet Movie Database) movie reviews. Each movie review within the dataset analyzed in this project was associated with having either a positive or negative sentiment. The objective of this project was to develop one or more models that could be accurately predict the sentiment of movie reviews as either positive or negative, based on analyzing the sentiment within the text of the IMDb movie review dataset. This task of sentiment analysis on the IMDb movie reviews was accomplished by the utilization of several deep learning models. The models implemented in this project included a word embedding model, SimpleRNN (Recurrent Neural Network) model, LSTM (Long short-term memory) RNN model, and a CNN (Convolutional Neural Network) model. The implementation of these models were done in the Python language through the *TensorFlow*and *Keras* packages. *TensorFlow* is a Python package developed by Google that is used to construct and train artificial neural networks through a low-level set of tools. *Keras* is a Python package that is able to be run in conjunction with *TensorFlow*. *Keras* is a very useful package for working with multiple types of artificial neural networks, including recurrent and convolutional neural networks.

The remainder of this paper will go into further detail on the steps involved in accomplishing sentiment analysis on the IMDb movie review dataset described above. This project was approached by initially collecting the necessary data, followed by extensively examining and preprocessing the data. The steps involved in the data preprocessing varied depending on the specific model being implemented. The word embedding, SimpleRNN, LSTM, and CNN models were each trained on the dataset, and the accuracy of their sentiment prediction was optimized through various neural network architectures.

1. **Literature review**

There have been several topics in the areas of text mining that provided background information for this project. One of these topics was mining of text data to predict consumer preferences.

While most of the literature in existence documents techniques around using reviews of movies, or movie taglines, for text extraction and sentiment analysis, the parallels with our approach were obvious and therefore it made sense to explore methodologies that have been successful in the past.

The two most common ways to structure computer readable text for measurement are using the bag-of-words approach and natural language processing (NLP). Specific to our challenge, the NLP approach aligns specific themes and emotions from movie scripts to map them into groups [2].

Sentimental analysis is classified into multiple sentimental classification techniques. Potential solutions that have been explored include Naïve Bayes classifiers, Support Vector Machines, Maximum Entropy Classifiers, Random forest, Recurrent Neural Network (RNNs) and Convolutional Neural Networks (CNNs).

At the heart of this type of problem are information extraction and the procedures, which were used to analyze resulting data sets that may contain a large number of predictors. Information extraction is a precursor to the analysis of text data and has a few notable techniques, which have been documented. Another topic of exploration was the classification and clustering of text data.

# Convolutional neural networks (CNNs) are well suited for text classification as it outperforms other models, such as bag-of-words (BOWs). CNNs proved its ability to learn automatically from scratch using character-level representations of text irrespective of the language used and without prior knowledge of language words, syntax, grammar and semantic similarities. CNNs allow high level understanding, provided the availability of enough data [6].

A recurrent neural network has a chain of repeating modules of neural network. shows the structure of typical RNN. Each block in the figure is called a module [8].

Long Short-Term Memory network is a special kind of recurrent neural network was proposed by Hochreiter & Schmidhuber [8] as an extension to recurrent neural network. LSTM networks are capable of learning long-term dependencies. It’s the default behavior of LSTM network to remember information for a long period.

Additionally, in recent years there were attempts to use clustering techniques which require less human input for training data have also been explored. Moreover, research specific to the film industry produced a couple of papers that provided additional background.

One paper described a proximity-based approach to text mining which was used to analyze movie reviews to classify them as either positive or negative. This method involved separating words into pairs and measuring the distance between them [3].

The second paper compared supervised and unsupervised learning methods of classification of movie reviews. A supervised learning method utilized N-gram 3 classifiers on documents that were represented as word tuples. This approach outperformed an unsupervised learning algorithm, which used a model that compared the similarity between phrases in the text and specific key words [10].

Classifiers on documents that were represented as word tuples. This approach outperformed an unsupervised learning algorithm, which used a model that compared the similarity between phrases in the text and specific key words.

A couple of other approaches to text mining outside of the entertainment industry were researched as well. One approach created an information distance from a document to a document cluster, which allowed text documents to be automatically summarized, and provided a means for classification. Another approach utilized a machine learning algorithm to build a classifier by learning from a set of training documents, in order to classify test documents [2, 5, 7].

Machine learning approach is based on text classification, which is used for forecasting and business decision making by automating the processing text. This approach is divided into supervised and unsupervised learning. The majority of practical machine learning uses supervised learning. In supervised learning approach, initially models are trained using classifiers of document. These trained models or documents have some key features, which have topic related words.

CloudScan trains a model that could be generalized to unseen invoice layouts, where a model is trained either using Long Short-Term Memory (LSTM) or Logistic Regression (LR) with expert designed rules as training features [10].

**CUTIE Model**

Convolutional Universal Text Information Extractor (CUTIE), applies convolutional neural networks on gridded texts where texts are embedded as features with semantical connotations. CUTIE tackles the key information extraction problem by applying convolutional deep learning model on the gridded texts.

1. **Methods**

The methodology for this study involved data collection, natural language processing, text data mining and the application of deep learning techniques for sentiment analysis. Text is one of the forms of sequence data. Text is either a sequence of characters or a sequence of words, but it is common to work with words. The deep learning models of sequence-processing used in this paper can use text to produce a basic form of natural-language understanding, enough for applications including document classification, sentiment analysis, author identification, and even question-answering. Also, it is critical to keep in mind that none of these deep learning models genuinely understand the text in a human intelligence; instead, these models can map the mathematical structure of written language, which is enough to solve many simple textual tasks.

The data used for this study is the large movie review dataset for binary sentiment classification. The core dataset contains 50,000 reviews split evenly into 25,000 train and 25,000 test sets. The distribution of labels is balanced 25,000 positives and 25,000 negatives. No more than 30 reviews were allowed for any given movie. In the labeled train/test sets, a negative review has a score <= 4 out of 10, and a positive review has a score >= 7 out of 10. Reviews with more neutral ratings were not included in the dataset.

**Working with deep learning models**

The first step in the analysis was to pre-process the movie review text data into a format that would support the utilization of the text data analysis functions with deep learning. Like all other neural networks, deep-learning models do not take as input raw text:

They only work with numeric tensors. *Vectorizing* text is the process of transforming the text into numeric tensors. There are many ways to vectorize the text:

* Use a unit of text into words and change each word into a vector.
* Use a unit of text into characters and change each character into a vector.
* Use n-grams of words or characters and change each n-gram into a vector. *N-grams* are overlapping groups of many successive words or characters.

Together, the different units into which we can break down the text are called *tokens*, and breaking text into tokens is called *tokenization*.

The process of vectorization consists of using some tokenization. Structure to associate numeric vectors with the tokens generated. These vectors, filled into the tensors, are feed into deep neural networks. There are numerous ways to associate a vector with a token. Here, we use the token embedding, called word embedding.

**Deep learning architectures used**

**Learning word embeddings with an embedding layer**

The easiest way to associate a dense vector with a word is to select the vector at random. The tricky with this method is that the resulting embedding space has no structure: for instance, the words *correct* and *right* may end up with entirely different embeddings, even though they are interchangeable in most sentences.

It is hard for a deep neural network to make an intelligent decision of such a noisy, unstructured embedding space. Because word embeddings are meant to map human language into a geometric space, for example, in practical embedding space, we would expect synonyms to be inset into similar word vectors; and usually, we would expect the geometric distance among any word vectors to relate to the semantic distance between the associated words. Also, in addition to distance, we may want specific *instructions* in the embedding space to be meaningful. Is there some ideal word-embedding space that would correctly map human language and could be used for any natural-language-processing task? Probably, but we have yet to compute anything of the sort. Also, there is no such thing as a *human language*—there are many different languages, and they are not similar in form, because a language is the reflection of a specific culture and a specific context.

Though, more pragmatically, what makes a great word-embedding space depends intensely on the specific task: the perfect word-embedding space for an English-language movie-review sentiment analysis the model may not look the same from the perfect embedding space for another English language, such legal-document-classification model, because of the importance of individual semantic relationships vary from task to task.

A picture containing crossword puzzle, text

Description automatically generated

Figure 1: Word embedding

**Understanding recurrent neural networks**

The primary characteristic of all neural networks, such as densely connected networks and convolutional neural networks, is that they have no memory. Each input shown to the networks is handled individualistically, with no state kept in between inputs. With such networks, in order to process a sequence of data points, you have to show the entire sequence to the network at once: turn it into a single data point. For instance, an entire movie review has to be transformed into a single large vector and processed in one process. Such networks are called *feedforward networks*.

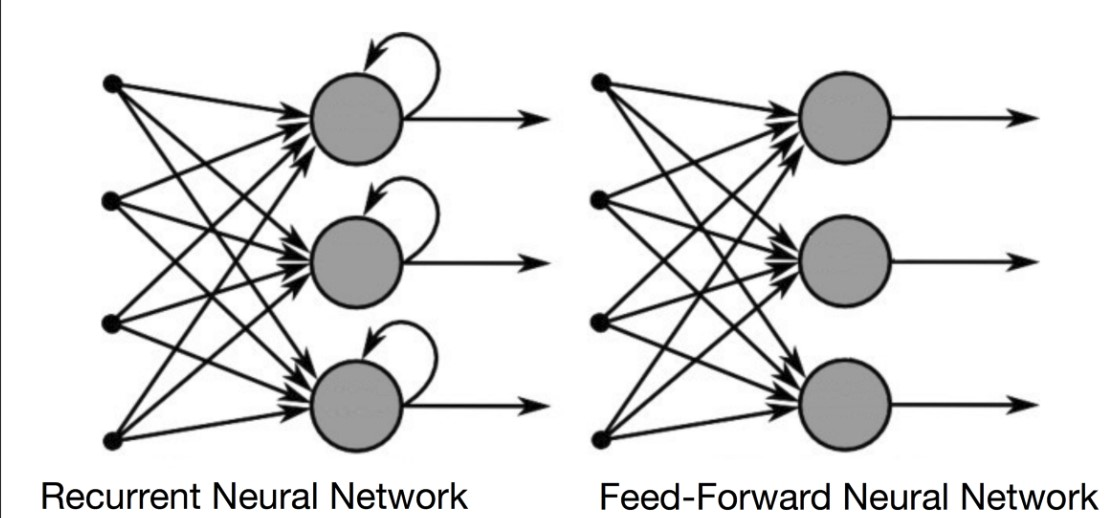


Figure 2: Recurrent Neural Network vs Feed-Forward Neural Network

However, when we as human, are reading a text paragraph or a sentence, we are processing it word by word while keeping memories of what came first; this gives us a good representation of the meaning conveyed by the text paragraph or sentence.

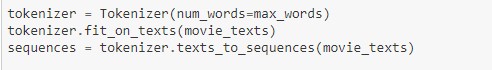


Figure : Text Tokenization

Our human intelligence processes information incrementally while maintaining an internal model of what it is processing, built from past information, and continuously updating as new information comes in.

A *recurrent neural network* (RNN), such LSTM, GRU, and SimpleRNN, adopts the same principle, although in a very basic form: it processes sequences by iterating through the sequence elements while preserving a *state* holding information relative to what it has seen so far.

**Text processing using convolutional neural networks**

Convolutional neural networks perform particularly well on computer vision problems, due to their ability to operate *convolutionally*, extracting features from local input and allowing for modular representation and data efficiency.

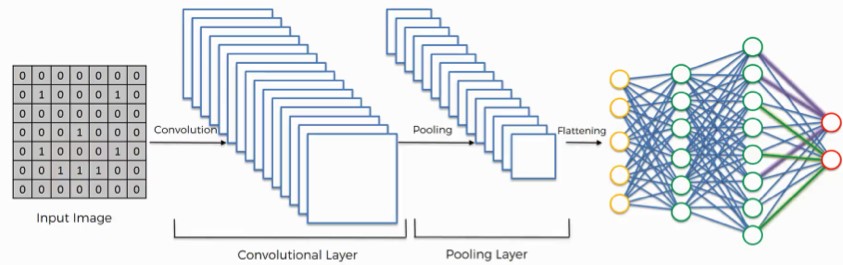


Figure 4: Convolutional Neural Network

The same properties also make them relevant to text sequence processing. Such 1D convolutional neural networks can be competitive with RNNs on some sequence-processing tasks, usually at a considerably inexpensive computational cost.

**The workflow**

Before training, we preprocessed the data by reshaping it into the shape the network expects and scaling it so that all values are in the [0, 1] interval.

Then, we fed the neural network the training data, train text data and train text labels. The network then learned to associate text and labels. Finally, we asked the network to produce predictions for test text data, and we verified whether these predictions match the labels from test text labels.

The core building block of neural networks is the layer, a data-processing module that you can think of as a filter for data. Some data goes in, and it comes out in a more useful form.

Specifically, layers extract representations out of the data fed into them —hopefully, representations that are more meaningful for the problem at hand. Most of deep learning consists of chaining together simple layers that will implement a form of progressive data refinement.

A screenshot of a cell phone

Description automatically generated

*Figure 5: Neural Network Model architecture*

To make the network ready for training, we need the following as, part of the compilation step:

A loss function—How the network will be able to measure its performance on the training data, and thus how it will be able to steer itself in the right direction. An optimizer—The mechanism through which the network will update itself based on the data it sees and its loss function. Metrics to monitor during training and testing

A screenshot of a cell phone

Description automatically generated

*Figure 6: Model compilation*

**Regularization techniques and tuning hyperparameters**

This step took us most the time. We had to repeatedly modify the models, train it, evaluated it on our validation data, modified it again, and repeat, until the models were good as it could get. Here are some of the techniques we used:

* Using different architecture,
* Add or remove layers
* Add L1/L2 regularization.
* Trying different number of units per layer of the optimizer.

1. **CRISP-DM**

The Cross-Industry Standards Process for Data Mining (CRISP-DM) was followed in our project. CRISP-DM is an industry standard guideline that companies followed when it comes to any type of data mining projects.

The life cycle model consists of six different phases with arrow showing the appendices between the phases. As the project is developed, the progress can move between development phases.

The first phase of the CRISP-DM model is Business Understanding. In this phase, we are trying to focus on understanding the objective we are trying to achieve, then come up with a problem that can be applied with data mining to solve.

The second phase of the CRIPS-DM model is Data Understanding, in this phase we are trying to understand what kind of data set that is available to us and how is this data set useful to the goal we are trying to achieve.

The third phase of the CRISP-DM model is Data Preparation. In this phase the raw dataset is modified to a structure that can be used to perform data mining on, this phase deals with the cleaning, reshaping, preparing and finally transforming the data into workable data set.

The database we used contains mostly text base, but since computer programs can’t understand text base data set, so we transform the data set into a data that our program can work with.

The fourth and fifth phase of the CRISP-DM model is Modeling and Evaluation, in this phase we apply different type of mathematic model towards our database and depends on the results we would select which model we would use to achieve the best results for the project’s objective.

The last phase is Deployment, in this phase where we select the best performing model to that can achieve our project goal and deploy the program.

1. **Results**

The solutions were evaluated using the train and test regimen on the following models on our text data set using 12,500 training samples:

1. Convolutional Neural Network (CNN) using 1D convnet implementation with ~90% accuracy and 81% validation accuracy.

A close up of a map

Description automatically generated

Figure 7: CNN - Training and validation accuracy

A close up of a map

Description automatically generated

Figure 8: CNN - Training and validation loss

2. SimpleRNN with 99% accuracy and 83% validation accuracy

A close up of a map

Description automatically generated

Figure 9: SimpleRNN - Training and validation accuracy

A close up of a map

Description automatically generated

Figure 10: SimpleRNN - Training and validation loss

3. Long-short memory (LSTM) with 97% accuracy and 86% validation accuracy

A close up of a map

Description automatically generated

Figure 11: LSTM - Training and validation accuracy

A close up of a map

Description automatically generated

Figure 12: LSTM - Training and validation loss

4. Word-Embedding using a pre-trained model with 97% and ~70% validation accuracy

A close up of a map

Description automatically generated

Figure 13: Word-embedding - Training and validation accuracy

A close up of a map

Description automatically generated

Figure 14: Word-embedding - Training and validation loss

5. Without Word-Embedding using a pre-trained model with 97% and ~82% validation accuracy

A screenshot of a cell phone

Description automatically generated

Figure 15: Without word-embedding - Training and validation accuracy

A close up of a map

Description automatically generated

Figure 16: Without word-embedding - Training and validation loss

**Additional models used:**

6. Reversed-order LSTM with 95% accuracy and 86% validation accuracy

7. Bidirectional RNN layer model with 97% accuracy and 85% validation accuracy

1. **Conclusion**

Three families of deep learning neural network architectures were used, densely connected networks, convolutional networks, and recurrent neural networks. Each type of neural network was meant for a specific input network architecture. The results of this study demonstrated that the use of deep learning techniques on movie reviews text data was a valid measure in measuring sentiment analysis.

This was evident in the fact that the average accuracy was 97% accuracy and 86% validation accuracy using LSTM. Validation accuracy with word embedding is 70% while the validation accuracy is 82% without word embedding. The validation accuracy is 83% using SimpleRNN.

In addition, deep learning techniques had some ability to effectively measure the positive and negative reviews into categories, as shown in this paper. While movies are typically classified based on genre, director or lead actors, this study was unique in demonstrating that movies can be analyze and potentially categorized based purely on the text from the rating reviews.

1. **Reference**
2. <http://ai.stanford.edu/~amaas/data/sentiment/>
3. F. Hu*et al*, "Emphasizing Essential Words for Sentiment Classification Based on Recurrent Neural Networks," *Journal of Computer Science and Technology,*vol. 32, *(4),*pp. 785-795, 2017.
4. A. Hassan and A. Mahmood, "Convolutional Recurrent Deep Learning Model for Sentence Classification," IEEE Access, vol. 6, pp. 13949-13957, 2018
5. Fei Hu Li Li Zi-Li Zhang Jing-Yuan Wang Xiao-Fei Xu, "Emphasizing Essential Words for Sentiment Classification Based on Recurrent Neural Networks,"vol. 32, *(4),*pp. 785-795, 2017.
6. Maas, Andrew L. et al. “Learning Word Vectors for Sentiment Analysis.” *ACL* (2011).
7. A. A. Helmy, Y. M. K. Omar and R. Hodhod, "An Innovative Word Encoding Method For Text Classification Using Convolutional Neural Network," 2019.
8. Johannes Michael, Roger Labahn, "Evaluating Sequence-to-Sequence Models for Handwritten Text Recognition" 2019
9. SUMESH KUAMR NAIR, RAVINDRA SONI, "Sentiment Analysis on Movie Reviews using Recurrent Neural Network," IRE Journals, vol. 1, 2019
10. J. Kaur, R. Dara and P. Matsakis, "Sentiment classification of short texts: Movie review case study," in 2018, . DOI: 10.1007/978-3-319-92058-0\_73.
11. Xiaohui Zhao et al., “CUTIE: Learning to Understand Documents with Convolutional Universal Text Information Extractor” 2019.