KF5042 Intelligent Systems

Cover sheet

Name	Mohamed Etri
User ID	W21015706
Course	Computer Science with artificial intelligence
Module Code	KF5042
Module Title	Intelligent Systems
Submission Date	
Word Count	

A Comparative Study of CNN, LSTM and SVM Models for Sentiment Analysis in Customer Reviews

Mohamed Etri (W21015706)

Department of Computer Science

Northumbria University

Newcastle Upon Tyne, United Kingdom

mohamedetri667@gmail.com

Abstract - This study compares three popular machine learning models, CNNs, LSTMs, and SVMs, for sentiment analysis in customer reviews from e-commerce platforms and review websites. The performance of these models is evaluated based on accuracy, precision, recall, and F1-score, along with the impact of data preprocessing, feature extraction, and hyperparameter tuning. Results indicate that LSTM models marginally outperform CNN models in handling long-range dependencies in text data, while SVMs demonstrate competitive performance. The study provides insights into the strengths, weaknesses, and trade-offs between these models for sentiment analysis tasks in customer reviews.

Keywords - Sentiment analysis, Convolutional Neural Networks (CNN), Long Short-Term Memory (LSTM), Support Vector Machines (SVM), Customer reviews, Model performance, Machine learning, Natural Language Processing (NLP)

I. INTRODUCTION

Sentiment analysis is crucial for businesses to understand customer opinions from product and service reviews. Recent advancements in AI, including deep learning methods like CNNs and LSTM models, have significantly improved sentiment analysis and text classification on large datasets.

SVMs have been frequently employed for sentiment analysis jobs in addition to deep neural networks. SVMs are especially effective when dealing with enormous datasets, as they can efficiently categorise data points into many classes. SVMs are well-known for their high performance in binary classification tasks, which makes them a good candidate for sentiment analysis.

The paper is organized as follows: Section II presents a literature review on machine learning and deep learning techniques used in sentiment analysis, including previous comparative studies and their findings; Section III describes the methods employed in this study, including the dataset(s) used, preprocessing steps, feature extraction, implementation of machine learning and deep learning techniques, and evaluation metrics; Section IV discusses the results and performance comparison of the techniques, along with trade-offs and limitations; Section V concludes the paper, summarizing the main findings, discussing implications for sentiment analysis in social media, and suggesting future research directions and potential improvements. Finally, Section VI lists all the sources cited in the paper.

II. RELATED WORK

A. Introduction to sentiment analysis

Understanding consumer attitudes and improving goods and services depend on sentiment analysis of customer evaluations. Deep neural networks have produced noteworthy outcomes in computer vision and text classification applications, particularly Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) models [1]. These methods entail teaching text representation and classifying text using trained features such the TF-IDF weighting scheme and the Parts of Speech (POS) [2].

Using datasets from Amazon food reviews, Twitter data, and restaurant evaluations, other research have investigated the applicability of sentiment

analysis in industries such as health, tourism, and telecommunications [3]. Comparative studies have assessed the efficacy of deep learning methods for sentiment analysis, including CNNs, RNNs, and GRU models [4]. Although local feature extraction by CNNs is effective, sequence correlations can be challenging [5]. However, LSTM models successfully combine sentiment and semantic representation, making them appropriate for jobs demanding more in-depth text comprehension [6].

Because of their ability to handle high-dimensional

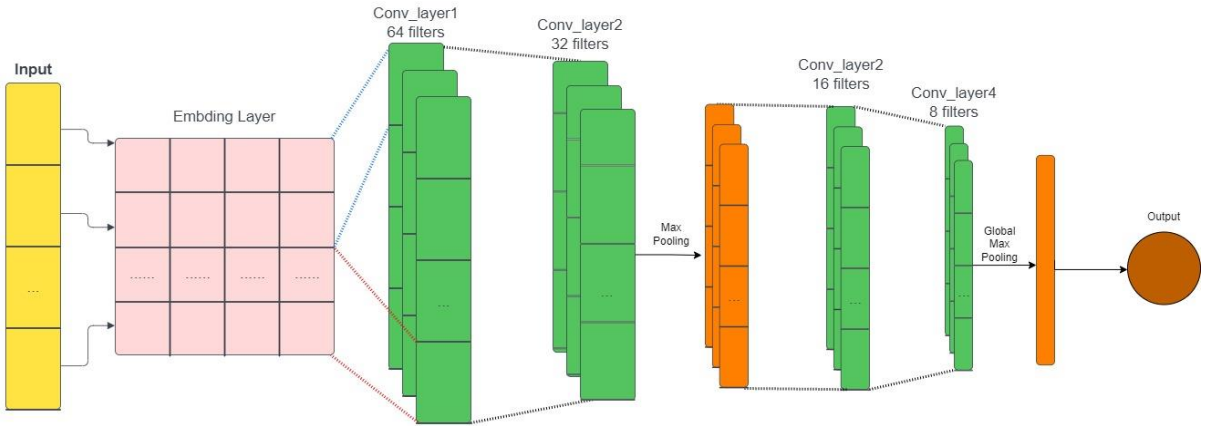

Figure 1. A convolutional Neural Network

feature spaces and non-linearly separable data, Support Vector Machines (SVMs) have also been utilised for sentiment analysis. On several datasets, SVMs have demonstrated promising results in text categorization and sentiment analysis applications. Furthermore, comparative studies have shown that SVMs outperform other machine learning models in sentiment analysis tasks, such as decision trees and Naive Bayes. SVMs, on the other hand, may require more feature engineering and hyperparameter adjustment than deep learning models to reach optimal results.

B. AI Techniques for Sentiment Analysis

Sentiment analysis has widely utilised AI algorithms to collect data from customer reviews, including support vector machines (SVM), convolutional neural networks (CNN), and recurrent neural networks (RNN), with long short-term memory (LSTM) models being a common RNN version. SVM is a popular supervised learning technique for binary and multi-class classification applications, including sentiment analysis. By mapping data into a high-dimensional space where it can be more easily separated by a hyperplane, SVM can handle high-dimensional feature spaces and non-linearly separable data. SVM can be computationally less expensive and require less training data than deep learning models for good performance.

CNNs and LSTMs, on the other hand, are deep learning models that can automatically learn to extract features from raw data and reach state-of-the-art performance in a variety of NLP applications, including sentiment analysis.

1) Convolutional neural networks

The input layer, where preprocessed text data is delivered into the network, is where the CNN architecture for sentiment analysis of customer reviews begins. Tokenizing the text data creates numerical representations of the text, such as word embeddings. Once the input has been tokenized, the embedding layer converts it into dense vectors of a given size, preserving the semantic context and minimising data dimensionality.

Four convolutional layers make up the CNN used for sentiment analysis, including two that use 64 and 32 filters to detect local patterns and a max pooling layer to preserve key features. Building on these features, the third and fourth layers (16 and 8 filters, respectively) uncover more complex patterns. In order to predict whether a review is favourable, negative, or neutral, an output layer uses a global max pooling layer to compress learned information into a vector. This CNN can anticipate the tone of upcoming, unread reviews, providing insightful information on consumer preferences.

Sentiment analysis in customer reviews leverages deep learning techniques, like Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) models, to effectively analyse and classify customer opinions. CNNs, known for their prowess in image recognition and natural language processing, excel in identifying local

features within text data [7]. However, CNNs may struggle to capture long-range dependencies and sequential correlations, limiting their context understanding [8], while LSTMs excel in processing sequential data and maintaining long-term dependencies [9].

2) Long Short-Term Memory (LSTM)

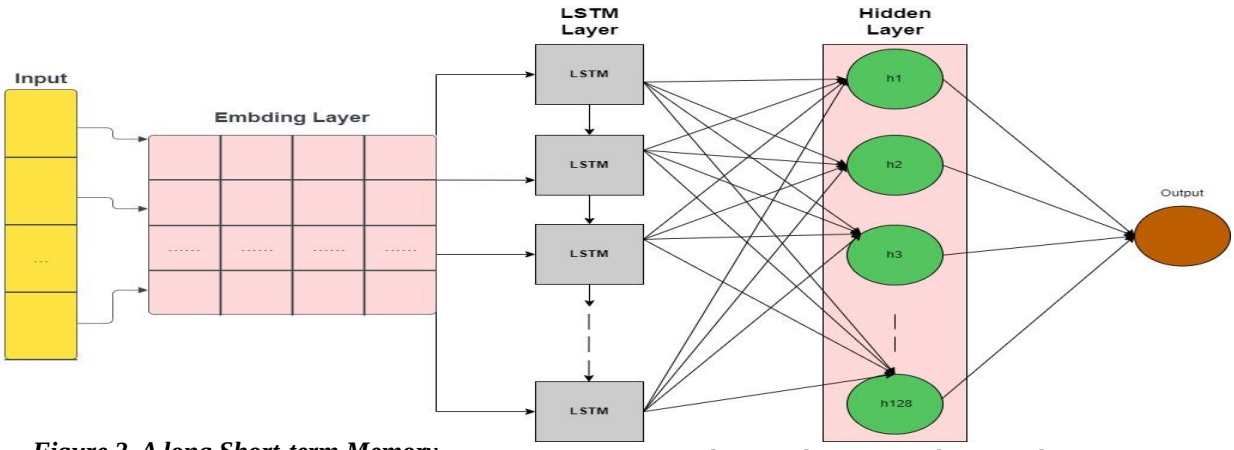

Figure 2. A long Short-term Memory

LSTMs, a specialized type of RNN, address the vanishing gradient problem associated with traditional RNNs by incorporating a memory cell [10]. This enables them to maintain and access long-term dependencies in input data. They have been widely used in sentiment analysis tasks, as they effectively model complex relationships between words, phrases, and sentence structures [11]. LSTMs are particularly suited for understanding sentiment in text data, as they capture temporal dynamics and contextual information in customer reviews.

An input layer, an embedding layer that converts the text into dense vectors using pre-trained word embeddings, an LSTM layer that extracts context and meaning, one or more hidden layers that further modify the data, and an output layer that forecasts sentiment by selecting the class with the highest probability are all components of the LSTM architecture for sentiment analysis in customer reviews. LSTMs have been shown to outperform other deep learning models, such as CNNs, in tasks involving capturing semantic relationships between words over longer sequences. However, they can be computationally expensive and may require more training time compared to CNNs [12]. When appropriately tuned and combined with techniques such as attention mechanisms and bidirectional architectures, LSTMs can yield state-of-the-art performance in sentiment analysis tasks [13].

3) Support Vector Machines (SVMs)

SVMs are a common machine learning technique used for binary classification applications such as sentiment analysis. They have been demonstrated to be effective in dealing with high-dimensional and sparse data sets, such as text data.

While SVMs perform well in some sentiment analysis applications, they suffer with highly imbalanced datasets and noisy data, resulting in poor generalisation and overfitting. Furthermore, they may fail to recognise the sequential pattern of text data, limiting their ability to discern context and nuances in customer evaluations.

C. Ethical, Social and Legal Issues in Sentiment Analysis

Sentiment analysis in customer reviews, while providing valuable insights, raises several ethical, social, and legal concerns that must be addressed to ensure responsible and equitable applications of AI technologies [14]. Two of the key issues that warrant further discussion are data privacy and algorithmic bias.

When processing customer feedback, data protection is a crucial factor. Users' privacy is protected by adhering to data protection laws like the General Data Protection Regulation (GDPR). Maintaining privacy standards depends on ensuring that personal data is anonymized and that it is stored and processed securely. To ensure transparency and ethical data practises, users' informed consent is required before any data is collected or used. To retain user confidence and legal compliance, researchers and practitioners must stay current with changing privacy rules and

put in place suitable data handling and security measures.

Another crucial issue in sentiment analysis is algorithmic bias. Biases existing in the training data may be inherited by AI models, including CNNs and LSTMs, resulting in unjust results. To reduce the chance of maintaining current prejudices, algorithmic bias must be addressed.

III. METHODS

A. Datasets

4000 reviews of Amazon Alexa and Echo devices are included in the "EcoPreprocessed.csv" dataset that was used in the research. It has four columns: a "review" column, a "polarity" column, a "division" column, and an index column. The "division" column contains the sentiment label of each review, which can be positive, negative, or neutral. The "review" column contains the text reviews of the devices, the "polarity" column contains the polarity score of each review, and the "division" column contains the polarity score of each review.

The dataset was subjected to data preprocessing and cleaning processes, which included deleting missing values, limiting the selection to the "review" and "division" columns, and categorising the sentiment labels. The sequences were padded to ensure equal length, and the text data was tokenized as well. By measuring the length of the longest sequence in the dataset, the maximum length was established. The hold-out approach with a 15% test size was then used to divide the dataset into training and testing sets.

B. Algorithms and Parameters

1) Convolutional neural networks

It has been demonstrated that convolutional neural networks (CNNs), a particular kind of neural network, excel at text classification tasks. An input layer, a convolutional layer, a max pooling layer, a fully connected layer, and a softmax output layer are all features of the CNN used in this work. The convolutional layer oversees extracting features from the text data, while the max pooling layer is utilised to minimise the output's dimensionality. Based on the retrieved characteristics, predictions are made using the fully connected layer and softmax output layer. In the convolutional and fully connected layers, the ReLU activation function is utilised.

2) Long Short-Term Memory (LSTM)

Recurrent neural networks, such as long short-term memory (LSTM) networks, excel in analysing sequential data, including text. By retaining a memory of prior inputs and selectively updating that memory at each time step, LSTMs are intended to capture long-term dependencies in sequential data. An input layer, an LSTM layer, a fully linked layer, and a softmax output layer are all present in this study's LSTM. With 'last' as its output mode, the LSTM layer has 100 hidden units. The CNN's use of the fully linked layer and softmax output layer is also applied here. A sequence of encoded text data with each element representing a word in the text serves as the input to the LSTM.

3) Support Vector Machines (SVMs)

Support vector machines (SVMs), a kind of linear classifier, operate by identifying the hyperplane that maximally divides the positive and negative samples in the feature space. The SVM in this study has a linear kernel, making it a linear classifier that operates by identifying the optimum hyperplane to distinguish between positive and negative samples. A bag-of-words representation of the text data, in which each document is represented as a vector of word frequencies, serves as the SVM's input. The SVM is trained using the training set before being applied to the test set to create predictions.

The neural network-based algorithms CNN and LSTM take different approaches to feature extraction. The LSTM retains a memory of past inputs to capture long-term dependencies in the text data, while the CNN executes convolution operations on the input text data to extract features. SVM, in contrast, is a linear classifier that does not use the text input to derive features.

The specific parameters used for the CNN and LSTM models are as follows:

- CNN: one convolutional layer with 128 filters and a kernel size of 5, one max pooling layer, one fully connected layer with 128 neurons, and a softmax output layer. ReLU activation function is used in the convolutional and fully connected layers.

- LSTM: one LSTM layer with 100 hidden units, a fully connected layer, and a softmax output layer.

In this study, sentiment analysis was also conducted using the SVM method with the scikit-learn library's default settings. Despite attempts at parameter customization, it was discovered that the default parameters produced the best outcomes. Finding a hyperplane that maximally segregates the positive and negative evaluations is how SVM operates in the context of sentiment analysis. SVM is still competitive with more complicated algorithms like CNN and LSTM despite its ease of use and speed.

C) Experimental Pipeline

The distribution of sentiments was preserved in both sets by splitting the dataset into training and testing sets with a ratio of 85:15.

The text input was tokenized and transformed into fixed-length sequences for the LSTM model using word encoding. A sequence input layer, an LSTM layer with 100 hidden units and output mode set to "last," a fully connected layer, a softmax layer, and a classification layer were used to build an LSTM network. The batch size was set to 64, and the Adam optimizer was employed with a learning rate of 0.001 and a maximum of 5 epochs. Accuracy was the performance metric, and the gradient threshold was set to 1. On the testing set, the LSTM model had an accuracy of 0.89. The same procedures were performed for the CNN algorithm however it only achieved an accuracy of 0.81.

Using the bag-of-words method, the text input was converted into numerical features for the SVM model. The training set was used to train an SVM model with a linear kernel, and the testing set was used to evaluate it. The performance metric employed was accuracy. On the testing set, the SVM model had an accuracy of 0.85.

D) Evaluation metrics

Accuracy, precision, recall, and F1 score are the evaluation measures employed in this sentiment analysis work (See Figures 3,4 and 5). These metrics were chosen because they offer a thorough assessment of the algorithm’s performances and are frequently employed in sentiment analysis tasks.

Accuracy measures the overall correctness of the predicted sentiment labels and is useful when the dataset is balanced. Precision and recall, on the other hand, provide insights into the model's ability to correctly predict positive sentiment labels and are useful when the cost of false positive or false negative predictions is high. Finally, the F1 score provides a balance between precision and recall, making it a good metric to use when there is an uneven class distribution in the dataset.

The following are the equations for calculating the evaluation metrics:

Accuracy = (True Positives + True Negatives) / Total Predictions

Precision = True Positives / (True Positives + False Positives)

Recall = True Positives / (True Positives + False Negatives)

F1 Score = 2 * ((Precision * Recall) / (Precision + Recall))

IV. RESULTS AND DISCUSSIONS

With an accuracy of 0.89 and an F1 score of 0.70, the LSTM algorithm has the best precision and recall ratios (see Figure 3). This indicates that compared to the other two models, the LSTM model is more accurate at correctly identifying positive instances and makes fewer errors when predicting positive instances. However, it might produce more false positives because it performs less accurately than SVM 0.66 (See Figure 5). This means that some instances that the LSTM model predicts as positive but are negative may occur. Nevertheless, the LSTM model outperforms the SVM algorithm with a recall score of 0.72, which indicates that it might overlook some true positives (See Figure 3).

The SVM algorithm has a precision score of 0.69, which is higher than the other two models and indicates that it generates fewer false positives. As a result, the SVM model is less likely to predict a positive instance when a negative one exists. It has an F1 score of 0.70, which is like the LSTM and shows that precision and recall are equally balanced (see Figure 5). The LSTM model can capture some true positive instances that the SVM algorithm may miss due to its lower recall score of 0.67 (See Figure 3). The accuracy score of the SVM algorithm, 0.87 (See Figure 3), is lower than the LSTM model's but still superior to the CNN models.

With an accuracy of 0.81 and an F1 score of 0.55, the CNN algorithm performs the least well of the three (see Figure 4). This indicates that compared to the LSTM and SVM models, the CNN model is less accurate at correctly identifying positive instances. The CNN model has a higher rate of false positives than the other two models, as evidenced by its precision score of 0.60 (See Figure 4). With a recall score of 0.55, it might be missing some true positive instances that the other models catch (See Figure 4). According to the CNN model's F1 score, precision and recall are not as evenly distributed as they are in the other two models, with precision having a higher weight than recall.

V. CONCLUSION AND FUTURE WORK

In conclusion, it can be said that the LSTM model outperforms the SVM and CNN models in terms of precision and recall ratios based on the analysis of the three AI algorithms. The LSTM model has a higher recall score than the SVM model, indicating that it captures more instances of true positives even though it may generate more false positives. The SVM model, on the other hand, has a higher precision score, which shows that it produces fewer false positives. With lower accuracy, precision, and recall scores than the other two models, the CNN model performs the least well. As a result of their superior performance, the LSTM and SVM models are suggested for use in this application. It is significant to note that if these models are used in real-world scenarios, there may be potential ethical, social, and legal issues related to predictive performance, which should be carefully considered.

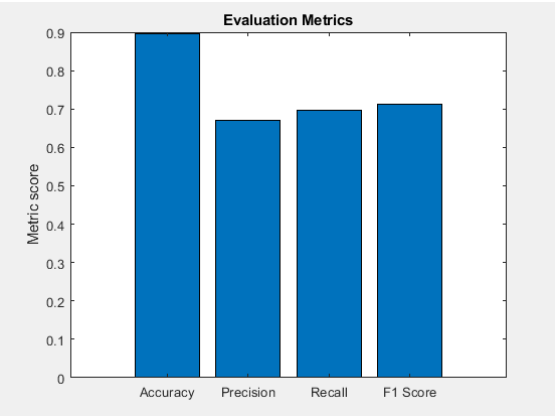

Figure 3. LSTM Evaluation Metrics

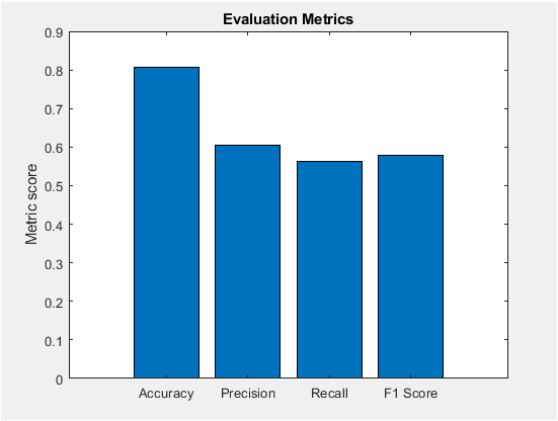

Figure 3. CNN Evaluation Metrics

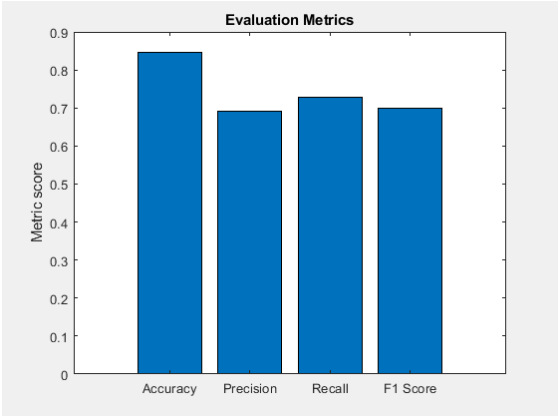

Figure 5. SVM Evaluation Metrics

VI. REFERENCES

1. Izhevsk, A., Sutskever, I. and Hinton, G.E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. [online] Neural Information Processing Systems. Available at: https://papers.nips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

2. Zhang, X. and Zheng, X. (2016). Comparison of Text Sentiment Analysis Based on Machine Learning. 2016 15th International Symposium on Parallel and Distributed Computing (ISPDC). [online] doi:https://doi.org/10.1109/ispdc.2016.39

3. Salas-Zárate, M. del P., Medina-Moreira, J., Lagos-Ortiz, K., Luna-Aveiga, H., Rodríguez-García, M.Á. and Valencia-García, R. (2017). Sentiment Analysis on Tweets about Diabetes: An Aspect-Level Approach. Computational and Mathematical Methods in Medicine, 2017, pp.1–9. doi:https://doi.org/10.1155/2017/5140631.

4. Do, H.H., Prasad, P., Maag, A. and Alsadoon, A. (2019). Deep Learning for Aspect-Based Sentiment Analysis: A Comparative Review. Expert Systems with Applications, 118, pp.272–299. doi:https://doi.org/10.1016/j.eswa.2018.10.003.

5. Tam, S., Said, R.B. and Tanriover, O.O. (2021). A ConvBiLSTM Deep Learning Model-Based Approach for Twitter Sentiment Classification. IEEE Access, 9, pp.41283–41293. doi:https://doi.org/10.1109/access.2021.3064830.

6. Gupta, U., Chatterjee, A., Srikanth, R. and Agrawal, P. (2017). A Sentiment-and-Semantics-Based Approach for Emotion Detection in Textual Conversations. [online] arXiv.org. Available at: https://arxiv.org/abs/1707.06996.

7. Zhang, Y. and Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. [online] arXiv.org. Available at: https://arxiv.org/abs/1510.03820.

8. Tang, D., Qin, B. and Liu, T. (2015). Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. [online] ACLWeb. doi:https://doi.org/10.18653/v1/D15-1167.

9. Severyn, A. and Moschitti, A. (2015). Learning to Rank Short Text Pairs with Convolutional Deep Neural Networks. Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR '15. [online] doi:https://doi.org/10.1145/2766462.2767738.

10. Tang, D., Qin, B. and Liu, T. (2015). Document Modeling with Gated Recurrent Neural Network for Sentiment Classification. [online] ACLWeb. doi:https://doi.org/10.18653/v1/D15-1167.

11. Tai, K.S., Socher, R. and Manning, C.D. (2015). Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers). doi:https://doi.org/10.3115/v1/p15-1150.

12. Zhang, Y. and Wallace, B. (2015). A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. [online] arXiv.org. Available at: https://arxiv.org/abs/1510.03820.

13. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł. and Polosukhin, I. (2017). Attention is All you Need. Advances in Neural Information Processing Systems, [online] 30, pp.5998–6008. Available at: https://papers.nips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

14. ETHI-CA2. (2016). Transparency in Machine Learning. LREC 2016 Workshop on Ethical Considerations in Human-in-the-Loop Systems. Retrieved from https://www.lrec-conf.org/proceedings/lrec2016/workshops/LREC2016Workshop-ETHICA2_Proceedings.pdf

Code:

Script 1. LSTM.m

```
% Load data
data = readtable('EcoPreprocessed.csv');
data = data(:, {'review', 'division'});
data = rmmissing(data);
numRows = size(data, 1);
sentiments = strings(numRows, 1);
for i = 1:numRows
    division = data.division(i);
    if strcmp(division, 'positive')
        sentiments(i) = "Positive";
    elseif strcmp(division, 'negative')
        sentiments(i) = "Negative";
    else
        sentiments(i) = "Neutral";
    end
end
data.Sentiment = categorical(sentiments);

% Tokenize data
num_words = 5000;
tokenizer = tokenizedDocument(data.review);
```

```
enc = wordEncoding(tokenizer);
X = doc2sequence(enc, tokenizer, 'Length', num_words);
max_length = max(cellfun(@(x) numel(x), X));
X = cellfun(@(x) [x, zeros(1, max_length - numel(x))], X, 'UniformOutput', false);
X = reshape(X, [1, numRows]);

% Create target variable
y = categorical(data.Sentiment);

% Split into training and testing sets
cv = cvpartition(numRows, 'HoldOut', 0.15);
idxTrain = cv.training;
idxTest = cv.test;
X_train = X(idxTrain);
X_test = X(idxTest);
y_train = y(idxTrain);
y_test = y(idxTest);

% Define LSTM network layers
input = 1;
numHiddenLayers = 100;
numClasses = 3;

layers = [
    sequenceInputLayer(input)
    lstmLayer(numHiddenLayers,'OutputMode','last')
    fullyConnectedLayer(numClasses)
    softmaxLayer
    classificationLayer];

% Specify training options
options = trainingOptions('adam', ...
    'MaxEpochs',4, ...
    'MiniBatchSize',64, ...
```

```
    'GradientThreshold',1, ...
    'Verbose',false, ...
    'Plots','training-progress', ...
    'ExecutionEnvironment', 'auto');

% Train the LSTM network
net = trainNetwork(X_train, y_train, layers, options);

% Evaluate the network
y_pred = classify(net, X_test);
accuracy = sum(y_pred == y_test) / numel(y_test);
disp(['Accuracy: ', num2str(accuracy)]);
```

Script 2: CNN.m

```
% Load data
data = readtable('EcoPreprocessed.csv');
data = data(:, {'review', 'division'});
data = rmmissing(data);
numRows = size(data, 1);
sentiments = strings(numRows, 1);
for i = 1:numRows
    division = data.division(i);
    if strcmp(division, 'positive')
        sentiments(i) = "Positive";
    elseif strcmp(division, 'negative')
        sentiments(i) = "Negative";
    else
        sentiments(i) = "Neutral";
    end
end
data.Sentiment = categorical(sentiments);

% Tokenize data
num_words = 5000;
tokenizer = tokenizedDocument(data.review);
```

```
enc = wordEncoding(tokenizer);
X = doc2sequence(enc, tokenizer, 'Length', num_words);
max_length = max(cellfun(@(x) numel(x), X));
X = cellfun(@(x) [x, zeros(1, max_length - numel(x))], X, 'UniformOutput', false);
X = reshape(X, [1, numRows]);

% Create target variable
y = categorical(data.Sentiment);

% Split into training and testing sets
cv = cvpartition(numRows, 'HoldOut', 0.15);
idxTrain = cv.training;
idxTest = cv.test;
X_train = X(idxTrain);
X_test = X(idxTest);
y_train = y(idxTrain);
y_test = y(idxTest);

% Define CNN network layers
inputSize = [1 num_words 1];
numFilters = 32;
filterSize = [3 3];
poolSize = [2 2];
numClasses = 2;

layers = [ imageInputLayer(inputSize) convolution2dLayer(filterSize, numFilters, 'Padding', 'same') reluLayer maxPooling2dLayer(poolSize, 'Stride', 2) convolution2dLayer(filterSize, numFilters*2, 'Padding', 'same') reluLayer maxPooling2dLayer(poolSize, 'Stride', 2) convolution2dLayer(filterSize, numFilters*4, 'Padding', 'same') reluLayer maxPooling2dLayer(poolSize, 'Stride', 2) fullyConnectedLayer(numClasses) softmaxLayer classificationLayer];

% Specify training options
options = trainingOptions('adam', ...
    'MaxEpochs', 4, ...
```

```
    'MiniBatchSize', 64, ...
    'ValidationData', {X_test, y_test}, ...
    'ValidationFrequency', 50, ...
    'Verbose', false, ...
    'Plots', 'training-progress', ...
    'ExecutionEnvironment', 'auto');

% Train the CNN network
net = trainNetwork(X_train, y_train, layers, options);

% Evaluate the network
y_pred = classify(net, X_test);
accuracy = sum(y_pred == y_test) / numel(y_test);
disp(['Accuracy: ', num2str(accuracy)]);
```

Script 3: SVM.m

```
function [accuracy, precision, recall, f1] = himate()

% Read in dataset
data = readtable('EcoPreprocessed.csv');

% Preprocess review text
docs = preprocessText(data.review);

% Convert division values to numerical labels
labels = zeros(size(data,1),1);
labels(strcmp(data.division,'positive')) = 1;
labels(strcmp(data.division,'neutral')) = 2;
labels(strcmp(data.division,'negative')) = 3;

% Split data into training and testing sets
cv = cvpartition(size(data,1),'HoldOut',0.2);
trainIDx = cv.training;
testIDx = cv.test;
trainDocs = docs(trainIDx);
```

```
trainLabels = labels(trainIDx);
testDocs = docs(testIDx);
testLabels = labels(testIDx);

% Create bag-of-words model
bags = bagOfWords(trainDocs);

% Remove infrequent words
bags = removeInfrequentWords(bags,5);

% Convert to matrix
XTrain = encode(bags,trainDocs);
XTest = encode(bags,testDocs);

% Train SVM model
svmModel = fitcecoc(XTrain,trainLabels);

% Test SVM model
predictions = predict(svmModel,XTest);

% Calculate evaluation metrics
accuracy = sum(predictions == testLabels) / numel(testLabels);

cm = confusionmat(testLabels, predictions);
precision = diag(cm)./sum(cm,1)';
recall = diag(cm)./sum(cm,2);
f1 = 2.*(precision.*recall)./(precision+recall);

% Plot evaluation metrics
figure
bar([accuracy, mean(precision), mean(recall), mean(f1)])
xticklabels({'Accuracy', 'Precision', 'Recall', 'F1 Score'})
ylabel('Metric score')
title('Evaluation Metrics')
```

```
end

function docs = preprocessText(textData)

% Tokenize text
docs = tokenizedDocument(textData);

% Remove stop words
docs = removeStopWords(docs);

% Stem words
docs = normalizeWords(docs,'Style','stem');

% Remove words with 2 or fewer characters
docs = removeShortWords(docs,2);

% Remove words with 15 or more characters
docs = removeLongWords(docs,15);

end
```

Link to Github repository:

https://github.com/moetri17/IntelligenceSystem