TOPICS IN ARTIFICIAL INTELLIGENCE (COMP - 8720)

PROJECT 2 - NATURAL LANGUAGE PROCESSING

Professor Name: Dr Robin Gras

Team:

Akshaya Muthuraman (Student ID: 110070509)

Ameer Asif Khader Batcha (Student ID: 110076465)

TABLE OF CONTENTS

INTRODUCTION

Sentiment analysis (also known as opinion mining) is a natural language processing (NLP) technique for determining the positivity, negativity, or neutrality of data. Sentiment analysis is frequently used on textual data to assist organizations in tracking brand and product sentiment in consumer feedback and better understanding customer demands. In this project, we analyze the yelp reviews dataset to classify the reviews into ratings ranging from 1 to 5.

YELP DATASET

The reviews.json file from the yelp dataset is used for analysis. This file is converted to a .csv file for easier processing. The file contains many columns like stars,text,date,funny,useful,cool, etc. An example of the dataset is shown below.

	review_id	user_id	business_id	stars	useful	funny	cool	text	date
0	KU_O5udG6zpxOg-VcAEod	mh_-eMZ6I	XQfwVwDr-	3	0	0	0	If you decide to eat here, just be aware it is going to take about 2 hours from beginning to end. We have tried it multiple times, because I want to like it! I have been to it's other locations in NJ and never had a bad experience. The food is good, but it takes a very long time to come out. The waitstaff is very young, but usually pleasant. We have just had too many experiences where we spent way too long waiting. We usually opt for another diner or restaurant on the weekends, in order to be done	#########

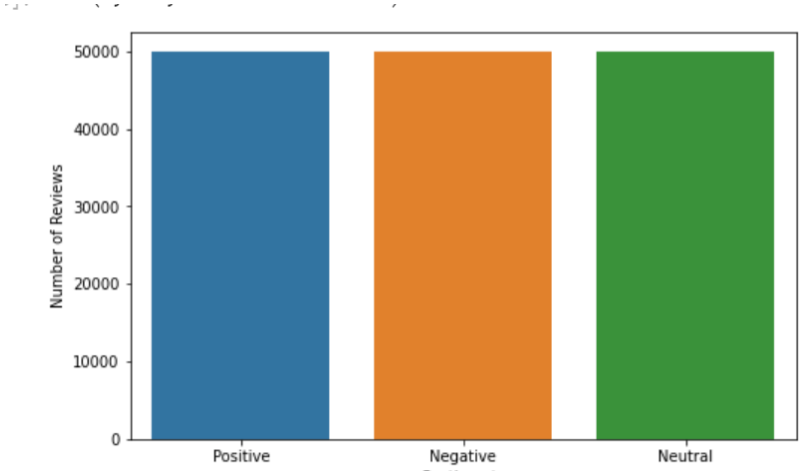
In this example, for a particular review, its related factors like the number of stars associated with the review, its usefulness is mentioned.

We take the text column that contains the yelp reviews as our input and the stars column as our output. This dataset is divided into training, validation and test datasets. Since this dataset contains approximately 7 million records, we take 1% of the data for training and 0.3% of data for testing.

DATA PREPROCESSING

There are various steps involved in data preprocessing. These steps are:

Step 1: The data is sampled and equal instances of each output label (Positive, Negative and Neutral) is retained.



Dataset division

Step 2: Dropping records with null values

Since these records do not contribute to improving the performance of the model, they are deleted.

Step 3: Remove stop words in English from the reviews.

Stop words are words that are typically filtered out before a natural language is processed. These are the most common words in any language (articles, prepositions, pronouns, conjunctions, and so on), and they don't add anything to the text. "The," "a," "an," "so," and "what" are some examples of stop words in English.
In any human language, there are plenty of stop words. We remove the low-level information from our text by deleting these terms, allowing us to focus more on the vital information.
Because there are fewer tokens involved in the training, removing stop words reduces the dataset size and hence reduces training time.

Step 4: Remove records that are not in the English language from the reviews

Since we are training our models with English reviews, all other language records are removed.

Steps 5: Remove punctuations from the reviews

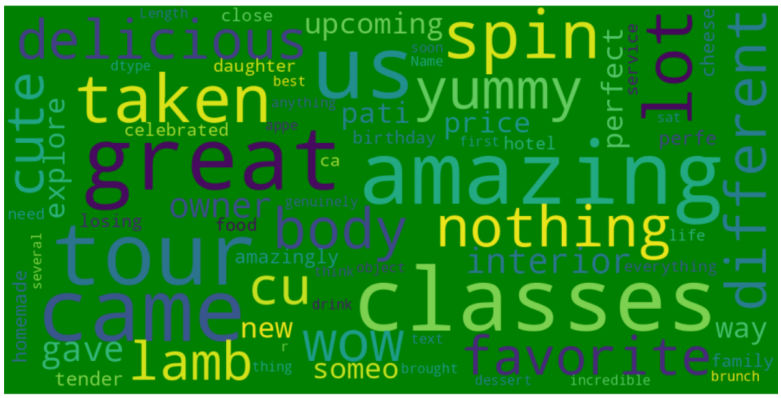
Punctuations do not contribute to the analysis of the sentiment of the review and can worsen its performance. Therefore, these punctuations are removed.

Step 6: Removes non-alphabetical characters from the reviews.

Step 7: Removes unnecessary words by keeping words only from large bodies of corpora from the reviews

Step 8: Vectorization (for basic LSTM model)
After processing, the textual data must be supplied into the model. This input must be vectorized because the model does not accept textual data and only understands numbers. The scikit-learn toolkit in Python has a great tool called CountVectorizer. It is used to convert a text into a vector-based on the frequency (count) of each word that appears throughout the text. This is useful when dealing with a large number of such texts and converting each word into a vector.

After preprocessing, a word cloud is used to represent the positive and negative words in the reviews as shown below.



Positive Words in the Review

Negative Words in the Review

Now we discuss the various models used in the project for the sentiment analysis of the preprocessed yelp dataset.

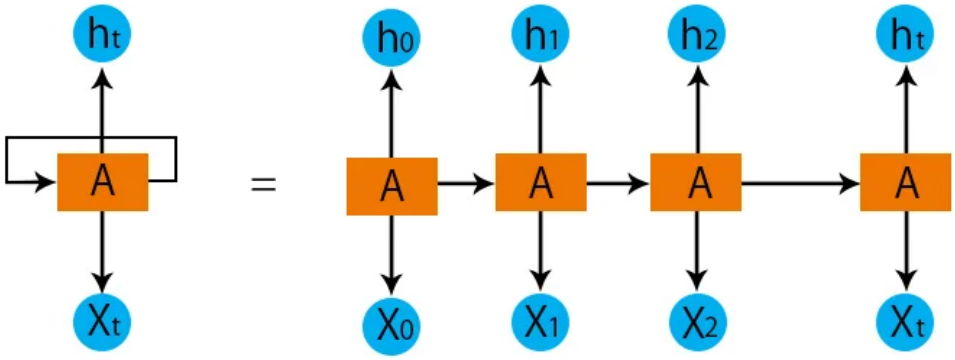
RECURRENT NEURAL NETWORKS(RNN)

A recurrent Neural Network is a generalization of a feed-forward neural network that has internal memory. RNN is recurrent in nature since it executes the same function for each data input, and the current input's outcome is dependent on the previous computation. The output is replicated and transmitted back into the recurrent network when it is created. It evaluates the current input as well as the output it has learned from the prior input when making a decision.
RNNs, unlike feed-forward neural networks, may process sequences of inputs using their internal state (memory). As a result, activities like unsegmented, connected handwriting recognition or speech recognition are possible. All of the inputs in other neural networks are independent of one another.

The steps in an RNN are[1]:

- The network is given a single time step of the input, i.e. xt is given to the network.
- The current state is then calculated using a combination of the current input and the previous state, i.e. For the next time step, the current ht becomes ht-1.
- We can go back as many time steps as the problem requires, combining data from all prior states.

- The final current state is used to determine the output yt once all of the time steps have been completed.
- After that, the output is compared to the actual output, and an error is calculated.
- The error is subsequently communicated back to the network, causing the weights to be updated and the network is trained.



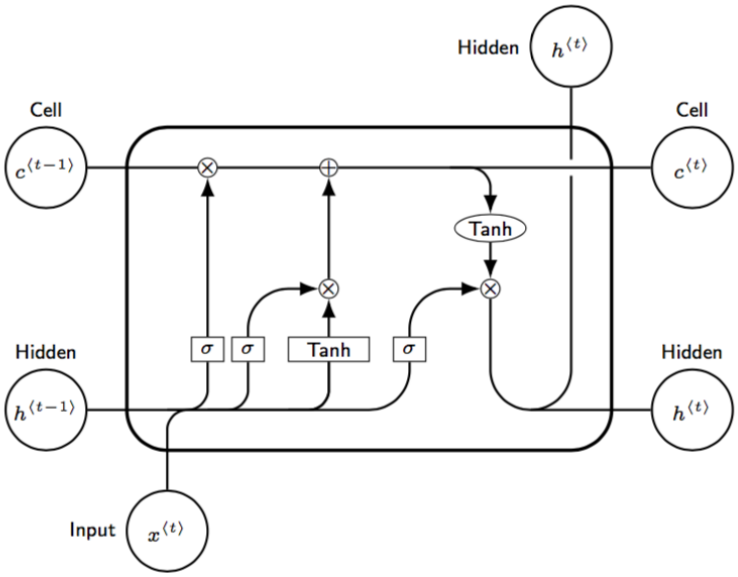
RNN Structure

LONG SHORT TERM MEMORY(LSTM)

We would be able to effectively use the RNN with a small RNN because there would be no problem with vanishing gradients. However, when using long RNNs, there isn't much we can do with typical RNNs, which is why they aren't generally implemented. That is what led to the discovery of LSTMs, which are fundamentally neuron structures that are slightly different. The main idea is that the gradients should not vanish even if the sequence is quite long.

A neuron will be referred to as a cell in LSTM. The only way a typical RNN can remember something is to update the hidden states and their weights. This challenge is overcome in LSTM by using an explicit memory unit to learn and remember tasks. It keeps track of information that is useful for learning.

It also employs a "Gating Mechanism," which regulates the information that the network keeps, determining whether it must send the information to the next layer or forget it[2].



LSTM Architecture

- A 'cell' that processes an input x(t) at time t, a previous hidden layer h(t-1) at time t, and a previous cell state c at time t. (t-1).
- At each given moment t, the cell produces two outputs: one is the output of the hidden state h(t), and the other is the output of the cell state c(t).
- The sigmoid function returns values between 0 and 1, whereas the tanh function returns values between -1 and 1. In LSTM, we shall employ these two basic activation functions.
- The sigmoid function takes in the input from x(t) and h(t-1) and we do a multiplication operation of it with the previous cell state c(t-1).
- 'Gate' is the name for this multiplication operation. If the sigmoid function returns a value near to 1, the multiplication will return a number close to c(t-1), implying that just a little portion of the previous memory is erased while the majority is retained. If the sigmoid function is close to 0, on the other hand, the multiplication will produce a number that is close to 0. This means you can almost completely wipe the previous cell state (memory). This is called '**Forget Gate**'.

- The next gate is named 'Update Gate,' and it utilizes a sigmoid and a tanh function, as well as a multiplication gate and an additional gate with output from the 'Forget Gate.' The 'tanh' function determines how much the value of the next cell state is increased or decreased. The sigmoid function determines how much data to write to the new cell state c. (t).
- 'Output Gate' is the next and final gate. A sigmoid function will be followed by a multiplication gate with a tanh activation function, releasing values to the hidden state for both the feed forward and recurrent sides. The greater the sigmoid function and tanh function values are, the higher the value transferred to the next hidden state h will be (t).

In LSTM, all three sigmoid and one tanh activation functions, whose input is a concatenation of h(t-1) and x(t), have different weights associated with them, namely w(f), w(i), w(c), and w(o). The total number of parameters required to train an LSTM model is four times that of a standard RNN. As a result, the computational cost is significantly larger. They devised a system known as GRU to address this issue.

GATED RECURRENT UNIT(GRU)

Unlike LSTM, it has only three gates and does not keep track of the Internal Cell State. The information stored in an LSTM recurrent unit's Internal Cell State is included in the Gated Recurrent Unit's hidden state. This aggregated data is forwarded to the next Gated Recurrent Unit. The following are the several gates of a GRU:-

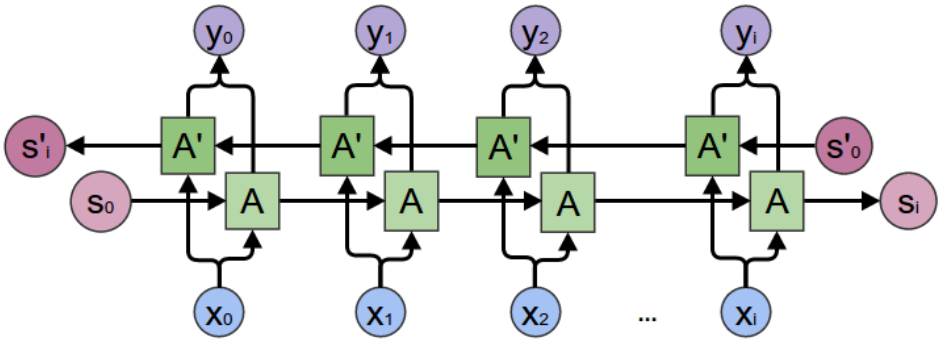
The Update Gate specifies how much past information must be passed on to the future. In an LSTM recurrent unit, it's similar to the Output Gate.

The Reset Gate specifies how much of the previous knowledge should be forgotten. It's similar to the combination Input Gate and the Forget Gate in an LSTM.

Current Memory Gate is a sub-component of the Reset Gate, much like the Input Modulation Gate is a sub-component of the Input Gate, and it is used to bring non-linearity into the input as well as make it Zero-mean. Another reason for making it a sub-component of the Reset gate is to reduce the impact of previous data on current data being sent into the future.

BIDIRECTIONAL RECURRENT NETWORKS(BRNN)

Bidirectional RNNs, or BRNNs, are used to provide straight (past) and reverse (future) input traversal. A BRNN is made up of two RNNs: one that goes forward, starting from the beginning of the data sequence, and the other that moves backwards, starting at the end. Simple RNNs, GRUs, or LSTMs can be used as network blocks in a BRNN. To support the backward training process, a BRNN features an additional hidden layer. The Back-Propagation Through Time (BPTT) algorithm is used to train a BRNN. In BPTT, At each time step, the network is unrolled and the errors are computed. Then the weights are updated and the network is rolled up.



Bidirectional RNN

BIDIRECTIONAL ENCODER REPRESENTATIONS FROM TRANSFORMERS (BERT)

BERT uses Transformer, which is an attention mechanism that learns contextual relations between words (or sub-words) in a text. This transformer consists of two mechanisms: an encoder that reads the text input and a decoder that generates a task prediction. The Transformer encoder reads the complete sequence of words at once, unlike directional models that read the text input sequentially (left-to-right or right-to-left). As a result, it is classified as bidirectional, however it is more correct to describe it as non-directional. This property enables the model to deduce the context of a word from its surroundings (left and right of the word).

Each input embedding is formed by three different embeddings:

Position Embeddings: To express the position of words in a sentence, BERT learns and employs positional embeddings. These are included to get over Transformer's constraint of not being able to record "sequence" or "order" information, which it lacks in comparison to an RNN.

Segment Embeddings: In addition to sentence pairs, BERT can accept sentence pairs as task inputs (Question-Answering).

Token Embeddings: These are the embeddings that the WordPiece token vocabulary has learned for a certain token.

Masked Language Modeling and Next Sentence Prediction are two NLP tasks that BERT has been pre-trained on.

ARCHITECTURES

We use 6 different models or architectures for sentiment analysis of the Yelp dataset. These are Simple RNN, LSTM, GRU, Bidirectional RNN, Bidirectional LSTM and BERT. The layers in these models are discussed here.

Embedding Layer

Textual data should be converted into numbers before feeding it into any machine learning model, including neural networks. If one-hot encoding is used, a dummy feature for each word will be created, which equals 10,000 features for a vocabulary of 10,000 words. This is not a practical embedding method because it necessitates a huge amount of storage space for the word vectors and decreases model efficiency. We can turn each word into a fixed-length vector of a specific size using the embedding layer. The resulting vector is dense, with real values rather than just 0s and 1s. The constant length of word vectors allows us to better represent words while reducing their dimensionality[5].

Dropout Layer

Dropout is a regularization technique that ignores a subset of neurons at random. They are "dropped-out" at random[6]. This means that on the forward pass, their

contribution to the activation of downstream neurons is removed temporally, and on the backward pass, no weight changes are applied to the neuron. As a result, the network's sensitivity to individual neuron weights decreases. Therefore, the network is more generalizable and less prone to overfitting the training data.

Dense Layer

A dense layer in a neural network is one that is deeply connected to the layer before it, implying that the layer's neurons are connected to every neuron in the layer before it. In artificial neural network networks, this layer is the most commonly used layer.

Optimizer

Adam is an optimization algorithm that can be used to replace stochastic gradient descent for deep training models. Adam combines the best features of the AdaGrad and RMSProp methods to create an optimization technique that handles sparse gradients on noisy problems.

Loss Function

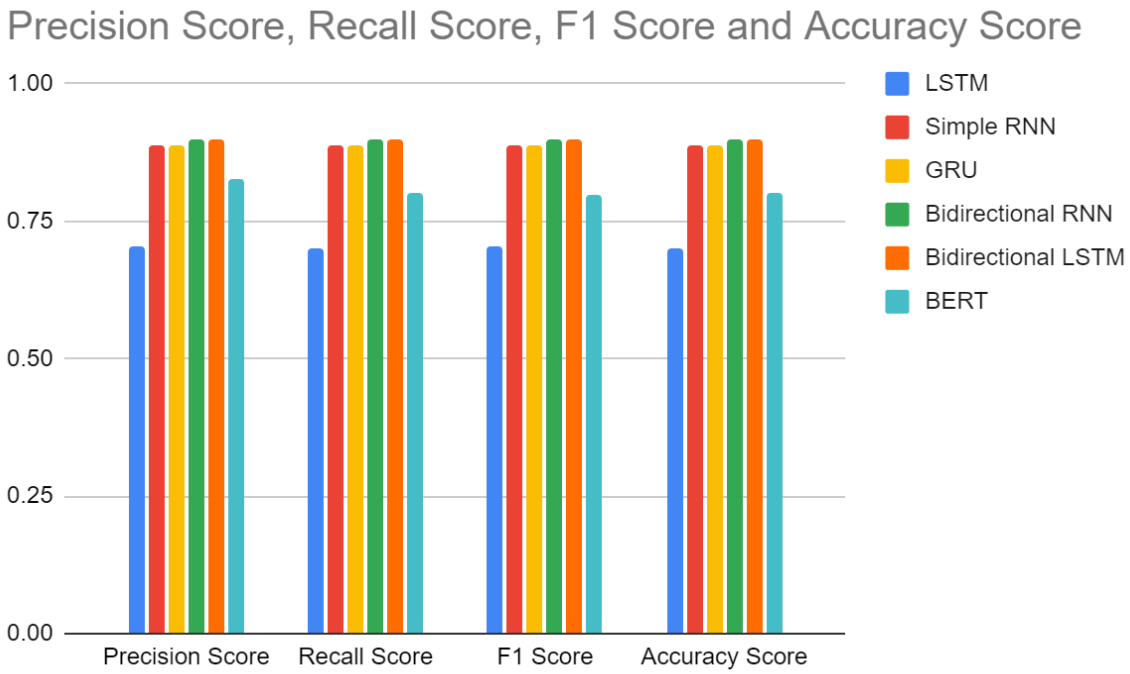
Categorical cross-entropy is used as the loss function throughout all 5 models as it is a multi-class classification (Positive, Neutral, Negative). One-hot encoding is done for the output labels.

PERFORMANCE ANALYSIS

The performance metrics are obtained from the confusion matrix of all the 6 models that are trained and tested. This gives the summary of prediction results on a classification problem. These metrics include accuracy score, precision score, recall score and f1 score.

For the simple LSTM, since the accuracy was low and overfitting occurred even after multiple attempts to rectify it, the 3-way classification of data (Positive, Neutral and Negative) is converted into a 2-way classification (Positive and Negative). This leads to much better results and accuracy as shown below.

	Precision Score	Recall Score	F1 Score	Accuracy Score
LSTM	0.7046	0.7013	0.7027	0.7017
Simple RNN	0.8871	0.8867	0.8867	0.8867
GRU	0.8871	0.8867	0.8867	0.8867
Bidirectional RNN	0.8971	0.8971	0.8971	0.8971
Bidirectional LSTM	0.8973	0.8971	0.8971	0.8971
BERT	0.8265	0.8008	0.7968	0.8008



From the results, we can see that Bidirectional RNN & LSTM works better for this dataset. Since a simple BERT architecture is used here, its performance is lesser than bidirectional RNN and LSTM. GRU and Simple RNN have better data preprocessing and thereby do better than LSTM. However, there is still some overfitting in these models.

CONCLUSION AND FUTURE SCOPE

In this project, we have used the Yelp dataset for sentiment analysis of reviews. Several sequence models like Simple RNN, LSTM, GRU, Bidirectional RNN, Bidirectional LSTM and BERT are used for this purpose and their results are compared. Only a small percentage of the dataset was used as it is computationally expensive. To improve performance, more data could be fed into the model and better overfitting control can be done. A more complex BERT model will lead to more accurate results. Currently the best results are obtained from Bidirectional RNN and LSTM.

REFERENCES

[1] Recurrent Neural Networks (RNN) | Working | Steps | Advantages (educba.com)

[2] Bi-directional RNN & Basics of LSTM and GRU | by Madhu Ramiah | Analytics Vidhya | Medium

[3] Microsoft Word - GRU_Variants_RahulDey_FathiMSalem_20Jan2017V.docx (arxiv.org)

[4] A Guide to Bidirectional RNNs With Keras | Paperspace Blog

[5] Understanding Embedding Layer in Keras | by sawan saxena | Analytics Vidhya | Medium

[6] Dropout Regularization in Deep Learning Models With Keras (machinelearningmastery.com)