COMP47650 Deep Learning
Final Assignment

Anonymous Author(s)
Affiliation
Address
email

Abstract

This project tackles one of the most difficult challenges in Machine Learning, Natural Language Processing. Using LSTMs and GRUs a very high level of accuracy was achieved on emotion classification on a Twitter tweets which is known to be a difficult text processing task due to the informal nature of the text.

1 Introduction

Automatic text classification has become a popular topic in machine learning. The aim of this assignment is to build a Deep Learning model using two different forms of Recurrent Neural Networks, LSTM(Long Short-Term Memory) and GRUs(Gated Recurrent Unit) to build a mode

2 Related Work

2.1 BERT

Transformers are the current state-of-the-art in Natural language processing. The paper "Attention is All you Need" was the paper that they stem from.

BERT is the most well known example of a transformer. It was pioneered by a team at Google AI Language. BERT stands for Bidirectional Encoder Representations from Transformers. It is described as being different from recent language representation models as it "is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers."[1] As a result of this, the pre-trained BERT representations can be fine tuned with just one additional output layer to create state-of-the-art models for a large number of tasks, for example question answering and language inference, without substantial task-specific architecture modifications. It uses an encoder and decoder layer to essential do two jobs: "fill in the blank" and "does sentence B come after sentence A?"

BERT-base was trained on 4 cloud TPUs for 4 days and BERT-large was trained on 16 TPUs for 4 days. Most people do not have access to this kind of computing power (and cannot afford it either), making the pre-trained paradigm very good at making a very accuracte NLP Model available to all.

Another advantage of BERT over LSTMs and GRUs is BERT's parralization capabilities. These models are inheritly sequetial in nature, precluding parallization within training examples. This can lead to extremely slow training times when we have a very large dataset with long sequence lengths as memory limits batching across examples.

It achieved state-of-the-art results for 11 NLP tasks including a GLUE score of 80.4%.

2.2 GPT-2

GPT-2 is a 1.5B parameter transformer that achieves state of the art results on 7 of 8 tested language modelling datasets. This language model was trained on a large and diverse data set: 45 million webpages curated/filtered by humans. This create the WebText dataset wwith 80 million rows of data. GPT-2 is a very large transformer model with 48 layers.

2.3 XLNet

XLNet was created by Carnegie Mellon University in conjunction with the Google Brain team. XLNet outperforms BERT on 20 tasks, often by a large margin[2]. XLNet is a generalized autoregressive pretraining method that utilizes the best auto-regressive language modelling(Transformer-XL) and autoencoding (BERT), while avoiding their limitations. It maximizes the expected log-liklihood of a sequence with respect to all possible permutations of the factorization order.

Due to the fact it is an autoregressive language model, XLNet does not rely on data corruption, and thus avoids BERT's limitations due to masking - i.e., pretrain-finetune discrepancy and the assumption that unmasked tokens are independent of each other.

2.4 T5

T5(Text-to-Text Transfer Transformer was developed by Google. It utilizes transfer learning, the method of which a model is first "pre-trained on a data rich task before being fine-tuned on a downstream task"[3]. The model is very large with 11B parameters. It has a GLUE score of 89.7 and a superGLUE score of 88.9 which almosts matches human performance (89.8).

3 Experimental Setup

This section includes a detailed description of the dataset used, the preprocessing applied to the dataset, the proposed algorithm(s), baseline approach and the hyperparameter tuning done.

3.1 Dataset Used

The dataset used for this assignment was (B.1) Text: "cleaned Emotion Extraction dataset from twitter" taken from Kaggle and is available here.

It contains 848,487 rows each containing the emotion of a tweet (disappointed, happy or angry) in the first column, while the second column is cleaned version of the raw tweet data in the third column.

The data is balanced with the proportion of disappointed, happy and angry being 34, 33 and 33 percent respectively.

Since this data is sentence based, this means that the data is sequential. This poses a problem for machine learning algorithms as the right architecture is essential. One of the main difficulty involving sequential data is machine learning algorithms, particularly neural networks, are designed to work with fixed length inputs. Furthermore, the temporal ordering of observations can make it difficult to extract features suitable for use as input to supervised learning models, meaning deep expertise in the domain is typically required.

One thing to note about this dataset is how the labels were generated. On the discussion tab of the Kaggle link above, the dataset creator say they used a model based on hashtags and emojis to label the data, meaning it was not done manually. Some tweets, therefore, may be not labelled exactly correct as these emojis or hashtags may have been used sarcastically.

3.2 Preprocessing Applied

Preprocessing is a very important part of any NLP task. There is a lot of clutter in languages, and this is especially true for twitter data which contains a lot of spelling mistakes and excess punctuation and other inconsistencies.

3.2.1 Punctuation and NAs

I removed any N/As from the dataset, as well as removing punctuation. While punctuation could be useful for a model like this, again it adds to the vocabulary size which would dramatically increase training time.

3.2.2 Number Removal

Numbers were removed from the dataset as they are generally not useful for emotion prediction.

3.2.3 Lower Casing

The text was converted to lowercase to reduce the amount of words in the vocabulary, otherwise, one word would be treated as two in the vocabulary, for example "Dog" and "dog" would be treated as different words due to one having a capital letter at the start. This would lead to a much larger vocabulary that would lead to a significant increase in training time.

3.2.4 Stop Words

Stop words are common words that appear very often in text data but hold low informational value to a language model. Words like "the", "and" and "it" are some examples. I removed stop words using the CountVectorizer from Scikit-Learn.

3.2.5 Stemming

I used the nltk package for stemming words. Stemming is the process of reducing inflection in a word into its root form. For example, connected and connection stemmed both become connect. The snowball stemmer from nltk is an effective package for stemming. "The 'english' (snowball) stemmer is better than the original 'porter' stemmer." according to the nltk website linked above.

Stemming is an effective way of dealing with sparsitiy issues. It also helps standardise vocabulary. it reduces down the size of the vocabulary which helps reduce training time.

3.2.6 Lemmatizing

I used the nltk package for lemmatizing. Lemmatizing is very similar to stemming, but is a more sophisticated approach. It transforms the word to the actual root using a dictionary such as WordNet for mappings. For example, the word "worse" would map to "bad".

Lemmatizing, does not appear to improve the accuarcy of the model significantly [1], however I have included it anyway as it did not add a significant amount of time to the overall modelling process.

3.2.7 Word Encoding

Word encoding involves assigning a number to each word. This makes it understandable by the machine learning algorithm as it cannot process strings as it does numbers.

3.2.8 Text Enrichment/ Augmentation

Word embedding is represents a word in a sentence as a real-valued vector. This vector encodes the meaning of the word such that words that are closer in the vector space are expected to be similar in meaning.

I originally used Word2Vec to train my own word embeddings on the data. While the dataset is quite large at approximately 900,00 rows, the model did not perform well using this word embeddings as there was not enough data. For my final model I used pretrained GloVe weights on 2Billion tweets for word embeddings.

GloVe stands for "Global Vectors", capturing both global statistics and local statistics of a corpus, in order to make the word vectors. GloVe has an advantage over Word2Vec in that it comes up with global statistics while Word2Vec only comes up with local statistics.

3.3 Proposed Algorithms

Two algorithms were used during this process: LSTM and GRU

3.3.1 LSTM

Long short-term memory is a type of Recurrent Neural Network. It is capable of learning order dependence in sequence prediction problems. LSTMs have a forget gate, an input gate and an output gate. These gates are fully connected layers with weights that can be trained using back propagation.

This enables the LSTM to regulate the flow of information as the gates decide which data is important and which data should be thrown away. By doing so, it can pass relevant information down the long chain of sequences to make predictions.

The use back propagation via the ADAM algorithm in order to optimize the parameters of these gates. The ADAM algorithm is an extension to stochastic gradient. The algorithm calculates an exponential moving average of the gradient and the squared gradient, and the parameters beta1 and beta2 control the decay rates of these moving averages.

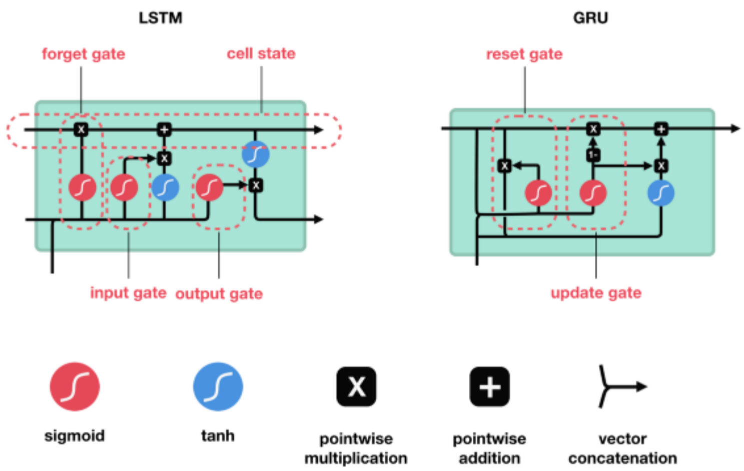

Figure 1: LSTM Architecture

3.3.2 GRU

GRUs are similar to LSTMs but have different gates. The only have an update and reset gate wheras the LSTM has the input, output and forget gate. This makes them faster to train.

3.4 Baseline Approach

For a baseline, I used an LSTM without word embeddings that general defaults for hyperparameters of 0.01 for the learning rate. As well as it being a single directional model. This had sub-standard results with accuracy of 70%.

3.5 Hyperparameter Tuning

Hyperparameter tuning is used to find the optimal model parameters that cannot be found using back-propagation or similar algorithms. After splitting the data into train, validation and test data, the model is trained on the train data and predictions are made using the validation dataset. Hyperparameters that gave the highest accuracy model are then were then chosen for the final model.

I used the ray tune package for hyper parameter tuning. It makes use of Async Hyperband Scheduling(ASHA) in order to increase efficiency. Hyperband "focuses on speeding up random search through adaptive resource allocation and early-stopping"[2].

It provides better parallelism and avoids straggler issues during eliminations compared to the vanilla Hyper Band Scheduler.

Figure X shows the ASHA Hyperband scheduler in action. We see there some trials are stopped early before 2, 4, 8 and 16(i.e powers of 2) iterations with the reduction factor parameter deciding how often a trial should be checked for early stopping. Some of the more successful trials, usally the ones ending up with 90 % accuracy, are run the full epoch amount. The accuracy plateaus around 8 - 14 epochs for the majority of the trials.

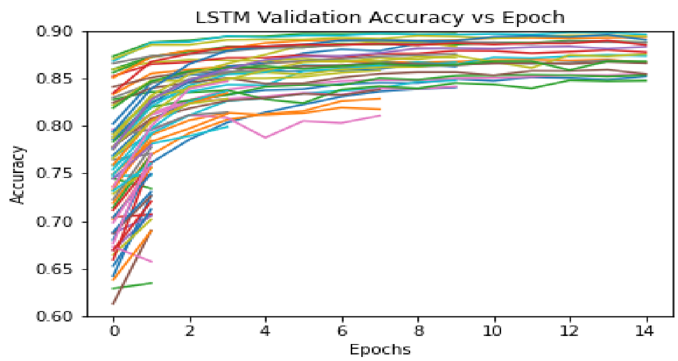

Figure 2: LSTM Hyperparamter Trials

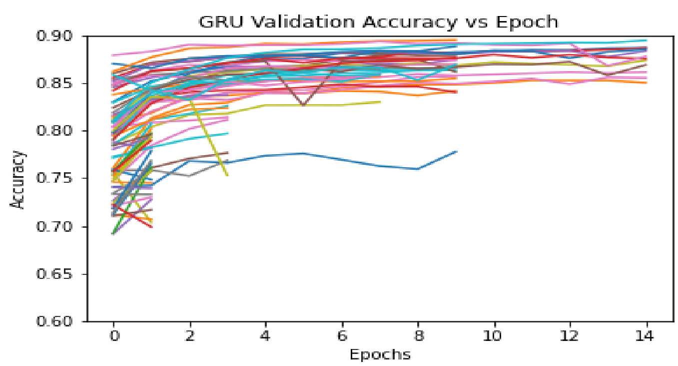

Figure 3: GRU Hyperparamter Trials

4 Results

Overall we see a very good accuracy on test data from both the LSTM and the GRU, with both having scores of just above 90%.

The F1 score on both models is also very high at approx 90%, indicating a good balance of precision and recall.

4.1 LSTM

We an see that the LSTM is very good at predicting between happy and angry, with almost no mistakes made. It has the greatest difficulty predicting Angry instead of Disappointed. This makes sense as you would imagine people using quite similar words to describe disappointment and anger while using completely different words to describe being happy.

Table 1: LSTM Results

Metric	Train %	Test %
Accuracy	89.67	90.15
F1-Score	89.79	90.20
Precision	90.01	90.52
Recall	89.72	90.19

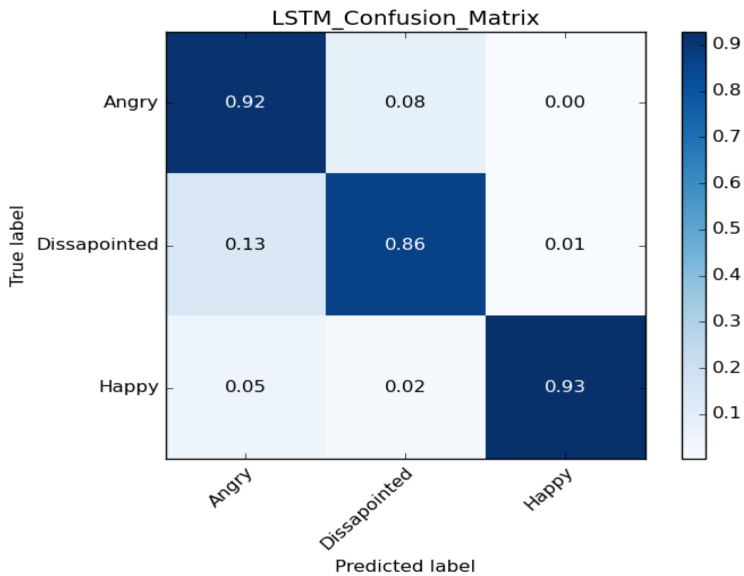

Figure 4: LSTM Confusion

4.2 GRU

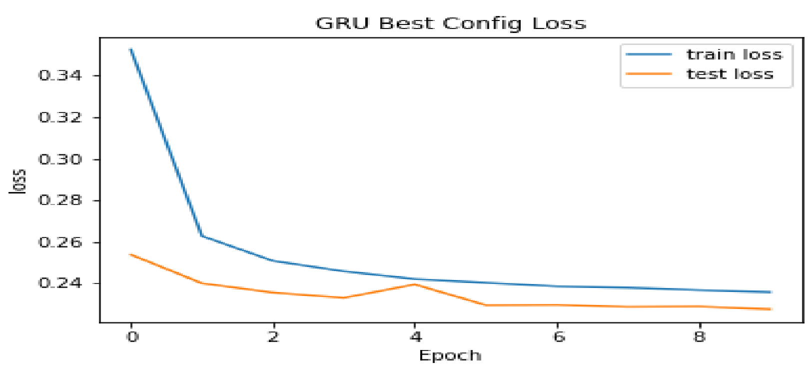

Figure 5: GRU loss per epoch

The GRU shows very similar performance to the LSTM, finding it most difficult to tell between a Disappointed and Angry tweet.

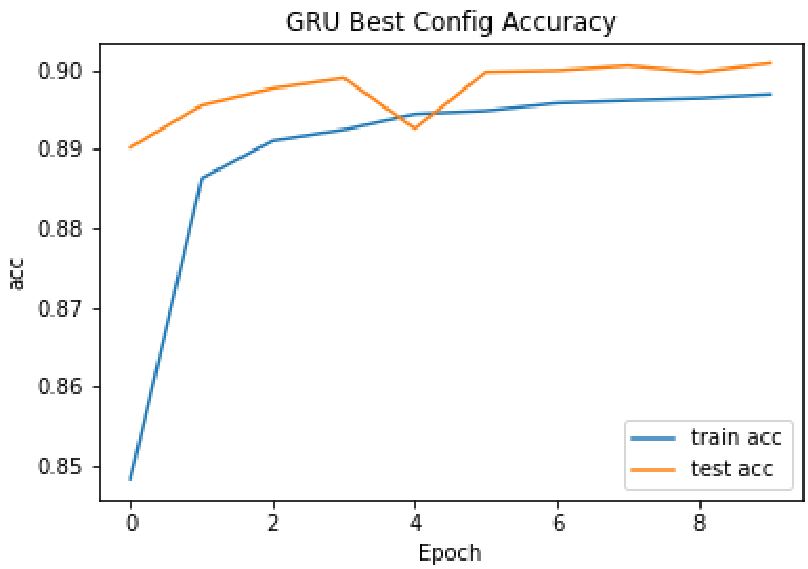

Figure 6: GRU Accuracy per epoch

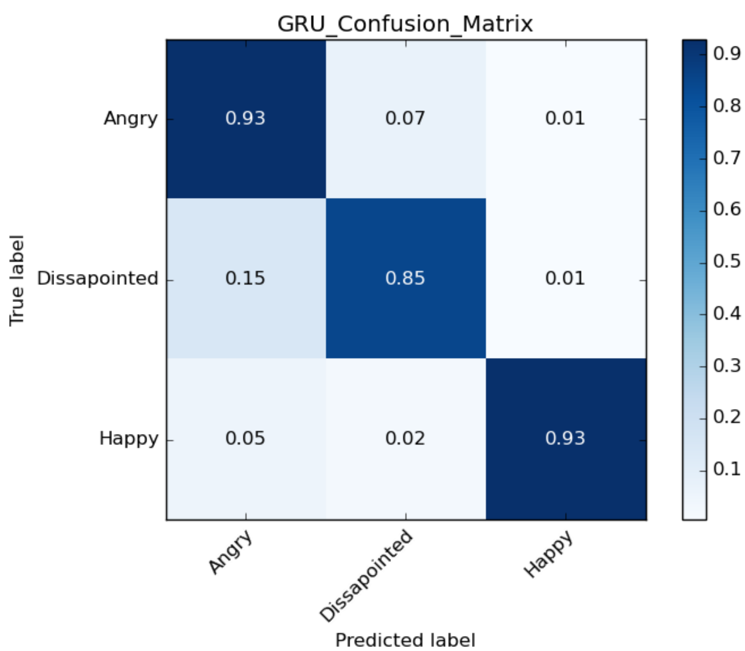

Figure 7: GRU Confusion

Table 2: GRU Results

Metric	Train %	Test %
Accuracy	89.70	90.01
F1-Score	89.81	90.08
Precision	90.09	90.56
Recall	89.75	90.08

5 Conclusion and Future Work

Both LSTMs and GRUs provide very good models for processing sequential data.

The accuracy of the models could be improved by expanding the search space of the hyper-parameters. The word embedding vector size used was 200, and this could be increased to 300 which could potentially lead to an increase in accuracy.

LSTMs and GRUs provide similar performance on this dataset. The GRU's training time is lower and hence a more attractive algorithm to use than the LSTM.

Future work could include building an ensemble of these two models along with additional models such as a convolutional neural network. Ensembles can lead to a reduction in bias an variance while also increasing accuracy.

We could build a model that does not use lowercase words only. People tend to use capital letters when expressing a strong emotion, hence including capital letters may lead to a more accurate model.

We could increase the maximum sequence length to include all 160 words. For my analysis, I used a max sequence length of 40 for the purpose of reducing training time. This obviously leads to the omission of some data which could have otherwise helped to develop a more accurate model. Including punctuation in the dataset should also be considered for a possible increase in accuracy.

Using other model architectures, especially pre-trained models such as BERT could potentially led to a more accurate model. Many transformers can achieve accuracy near 99% when trained with enough data.

References

[1] Vaswani, A. & Shazeer, N.M. (2017) Attention is All you Need

[2] Yang, Z. & Zihang, D. (2020) XLNet: Generalized Autoregressive Pretraining for Language Understanding

[3] Raffel, C. & Shazeer, N. (2019) Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer

[4] Camacho-Collados, J. & Pilehvar, M.T. (2018) On the Role of Text Preprocessing in Neural Network Architechtures: An Evaluation Study on Text Categorization and Sentiment Analysis. In *Journal of Artificial Intelligence Research (JAIR) 7*, pp. 2. Cambridge, MA: MIT Press.

[5] Li, L. & Jamieson K. (2018) Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization *Journal of Machine Learning Research 18* Cornell University

A Appendix