Hindi Tweets Sentiment Analysis Using Transfer Learning

Pre Ph.D. Course Work
Dissertation

In

Computer Science

by

Anubhav Mehra

Under the supervision of
Dr. Manoj Kumar Bisht

DEPARTMENT OF COMPUTER SCIENCE
S.S.J. UNIVERSITY
ALMORA - 263601(INDIA)
JULY, 2021

ACKNOWLEDGMENT

It is a genuine pleasure to express my deep sense of gratitude to my mentor and my Ph.D. supervisor **Dr. Manoj Kumar Bisht**. His advice and approach have helped me to a very great extent to accomplish this task.

I would also like to extend my gratitude to the Department of Computer Science, S.S.J University, Almora HOD ma'am **Dr. Parul Saxena** and all other faculty members who have been of great help during the course of this project.

Finally I would like to thanks my head **Dr. Ashish Mehta**, Department of Computer Science, DSB Campus Nainital for his immense help and providing me with time to complete my dissertation work.

DECLARATION

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all the principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause of disciplinary action by the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Anubhav Mehra
July 29, 2021

Department of Computer Science
Soban Singh Jeena University, Almora,
Uttarakhand

CERTIFICATE

I hereby certify that the work which is being presented in the dissertation entitled "**Hindi Tweets Sentiment Analysis using Transfer Learning**" submitted in the Department of Computer Science of the Soban Singh Jeena University, Almora is an authentic record of my own work carried out under the supervision of Dr. Manoj Kumar Bisht, Soban Singh University Almora.

The matter presented in this dissertation has not been submitted by me for the award of any other degree of this or any other University/Institution.

(Anubhav Mehra)
Candidate

(Dr. Manoj Kumar Bisht)
Supervisor

Date: _____

ABSTRACT

Sentiment analysis is a **natural language processing** technique to find if the sentiment of the text is positive, neutral or negative. Traditionally, to train a model for sentiment analysis require very dense neural networks to train on very huge datasets. But, here we have used a technique called **Transfer Learning** that stores a model which has learned some knowledge, that we can leverage in solving some other tasks based on the knowledge of the previous model. Here we are using a bidirectional pre-trained language model called **BERT(Bidirectional Encoder Representations from Transformers)**. BERT is a pertained model which learns using the learning techniques developed by Google. The BERT multilingual base model that we are using is pertained on the top 104 languages including Hindi. We then leverage the power of this model for the Sentiment analysis of the Hindi texts dataset that we've got. This allows us to achieve moderately high accuracy scores using a comparatively small dataset.

Keywords: Sentiment Analysis, Emotion AI, Natural Language Processing, Transfer Learning, BERT, Multilingual Natural Language Processing, Hindi Sentiment Analysis.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Transfer Learning

Transfer Learning is a Machine Learning method where a model that is trained for a certain task is utilized as the starting point for solving some other task i.e., to train a second model from the knowledge learned from the first model as well as the dataset. It is a very popular approach in natural language processing domain to solve problems such as getting the context of the text.

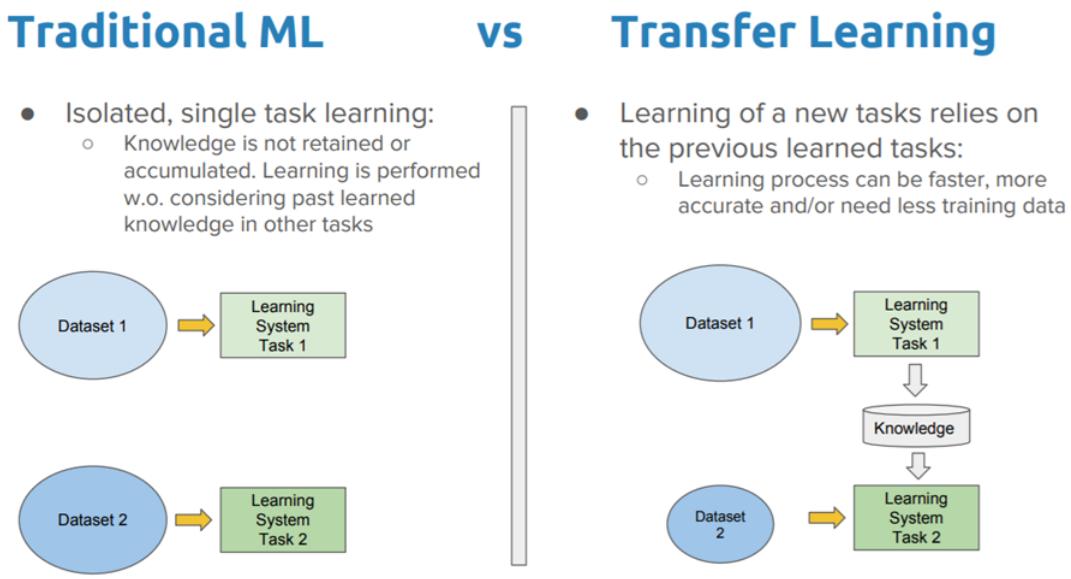


Figure 1: Differentiation between Traditional and Transfer Learning Methodology.[1]

The inspiration for transfer learning comes from us - humans, ourselves - where in, we have an inherent ability to not learn everything from scratch. We transfer and leverage our knowledge from what we have learn in the past for tackling a wide variety of tasks.[2]

Deciding when to use Transfer Learning

As with any technique at our disposal we must have a very clear reason on using transfer learning. First and foremost, we must realize that we can't use transfer learning in every situation. This is because for transfer learning to work we must have a model trained on a similar task. As with the problem concerned in our dissertation here, we have a language model available which is trained in recognizing linguistic structure of various languages. Now, we can leverage the power of this model to train a new model for our

context analysis problem. If this is the case, you will realize that transfer learning is a very decent technique to use for acquiring better results and that too in very short amount of time than traditional methods. Now, with this out of the way you may want to use transfer learning if you have following conditions:

- You have a smaller dataset
- You are constrained on time

Applications of Transfer Learning

There are various applications of Transfer Learning in variety of domains. In **Image Recognition**, transfer learning can be used to perform various imaging tasks. For e.g., a model trained to identify face can be used for facial recognition. Transfer Learning can also be used in **Speech Recognition** tasks. For e.g., a model trained in a particular language for speech recognition can be used to train a model for speech authentication. But our key domain of interest where we are most interested in using this technique is **Natural Language Processing**. Transfer learning can be used to solve various tasks in NLP. For e.g., a model trained in recognizing the linguistic structure of a language can be used to train a model to predict next word based on the series of words it gets in the previous sentences.

1.2 Sentiment Analysis

One such task in NLP is **Sentiment Analysis**. Sentiment Analysis or emotion AI, is the process in natural language processing of subjective emotional analysis of the text. Primarily sentiment analysis finds if the emotional tone of a piece of writing is ***positive, neutral or negative***. **Table 1** lists some examples of what a sentiment analysis categorization may look like.

Table 1: Sentiment Analysis Categorization.

Text	Category
That restaurant has a great food	Positive
He is my brother's colleague	Neutral
Bollywood movies are not entertaining	Negative

As we can see it is easily understood by a human brain what sentiments these pieces of

writing represent. But, for a computer this is a very challenging problem. It is a challenging problem because the way humans communicate using natural languages is extremely varied depending upon *subjectivity*, *use of irony*, and the *context* of the phrase. Not all verbs, nouns and pronouns can be treated equally when analyzing the emotional tone of the text. The tone of our phrases depends on how we are using them, when we are using them and for what we are using them. For e.g., the phrase *absolutely everything*, can be an answer to a question like *What are you so happy about?* or can equally be an answer to a completely opposite question *What are you so sad about?*. Thus, the emotional tone of the phrase can have completely separate meaning depending upon the various parameters. In the example above, the context of the phrase can alter the emotional meaning of the phrase completely.

Applications of Sentiment Analysis

Listed below are some applications of Sentiment Analysis:

- **Social Media Monitoring:** Sentiment analysis is very useful tool to find out what is a general sentiment of people in social media towards a product, person, situation, trend or any such topic of interest.
- **Feedback Analysis:** Sentiment analysis can also be used to find the whether the reception of new policies or workings of a company or of a government is positive or negative. This helps a company or government to evaluate their policies and performance using a well-defined metric.
- **Market Research:** Sentiment analysis can also be used to find what topics are viewed in positive and negative light in current landscape. This information can then be used to design or steer a companies marketing campaign to align with those sentiments.

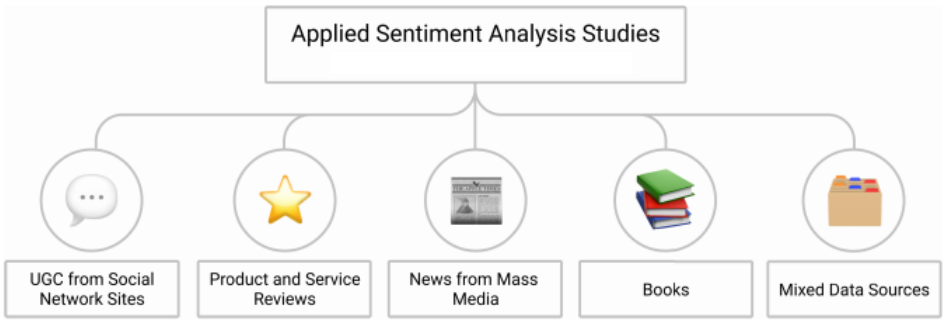


Figure 2: Application of Sentiment Analysis.[3]

1.3 Purpose

The purpose of this project is to showcase the use of Transfer Learning in developing a model for Sentiment Analysis of Social Media Posts(tweets in this case) posted in Hindi Language. Thus, calculating the accuracy of the model trained on a relatively small dataset and quick training time compared to those that would have been required if we would've followed a Traditional ML route. Hence, contributing in the field of Machine Learning and Data Science for any future project work in technology development.

1.4 Dissertation Structure

Chapter 2: This chapter discusses the literature review of this project. It discusses the concepts of this project, related an similar works done.

Chapter 3: This section describes the steps of the method followed to complete the project, which includes data sourcing, data preprocessing, model selection, model implementation, accuracy score calculation.

Chapter 4: This section describes the result and conclusion of the project.

Chapter 5: This section details what further works and improvements can be incorporated in the project.

Chapter 6: This section contains the bibliography for this dissertation.

2 Review of Literature

This section details various concepts behind this project, their workings and also similar types of works that have been done.

2.1 Concept of Transfer Learning

2.1.1 General Idea

While, we have new and better techniques of developing deep neural networks that are astoundingly great at predicting outputs for particular features set. We still have a problem at hand. What our models still frightfully lack is the ability to generalize to conditions that are different from the ones encountered during training. When is this necessary? Every time you apply your model not to a carefully constructed dataset but to the real world. The real world is messy and contains an infinite number of novel scenarios, many of which your model has not encountered during training and for which it is in turn ill-prepared to make predictions.[4]

So to make a generalizing model the dataset required that would enable the model to learn the working in the real world would be astoundingly huge. Collecting such dataset may be difficult, expensive and can also be downright impossible in many cases. So for a solution we turns our head towards transfer learning.

Humans are generally very good at transfer learning. We use our knowledge at one task to solve a related task all the time. For example, if you have any experience programming with Java, C/C++ or any other programming languages, you're already familiar with concepts like loops, recursion, objects and etc (Figure 3, a). If you then try to pick up a new programming language like python, you don't need to learn these concepts again, you just need to learn the corresponding syntax. Or to take another example, if you have played table tennis a lot it will help you learn tennis faster as the strategies in these games are similar (Figure 3, b).

a) Transferring Learned knowledge from Java to Python.

b) Transferring Learned knowledge Table Tennis to Tennis.

Figure 3: Transferring the learned knowledge in humans.

In Transfer Learning as compared to Traditional Learning where models are trained in complete isolation, we use the knowledge gained by a model in solving a task of similar domain to train our desired model to solve some other task.

2.1.2 Definition of Transfer Learning

Following is the definition of Transfer Learning in terms of domain and tasks.

A domain D consists of: a feature space X and a marginal probability distribution $P(X), where\ X = \{x1, ..., xn\} \in X$. Give a specific domain, $D = \{X, P(X)\}$, a task consists of two components: a label space Y and an objective function $f : X \rightarrow Y$. The function f is used to predict the corresponding label $f(x)$ of a new instance x. This task, denoted by $T = \{Y, f(x)\}$, is learned from the training data consists of pairs $\{x_i, y_i\}$, where $x_i \in X$ and $y_i \in Y$.[5]

Give a source dome D_S and learning Task T_S, a target domain D_T and learning task T_T, where $D_S \neq D_T$, or $T_S \neq T_T$, transfer learning aims to help improve the learning of the target predictive function $f_T(\bullet)$ *in* D_T using the knowledge D_S and D_T.[5]

2.2 Sentiment Analysis using Transfer Learning

The current landscape of sentiment analysis most sentiment analysis methods are classified from two perspectives. One is according to the granularity of text analysis, which is divided into three aspects: document level, statement level and aspect level. The other is according to the principle of method, it is mainly divided into three types: rule-based, machine learning, and deep learning. However, with the increasing volume of information, the collected data that need to be analyzed are huge, chaotic and irregular. This has brought more difficulties for sentiment analysis as large data labeling and high cost of computing. Although the traditional sentiment analysis methods have some advantages, they are more limited due to massive data requirements in practical applications. Therefore, researchers have combined sentiment analysis and transfer learning.[6]

2.2.1 Language Model Pre-Training

The intuition behind pre-trained language models is to create a black box which understands the language and can then be asked to do any specific task in that language. The language model is first fed a large amount of annotated data (for example, the complete Wikipedia dump). This lets the model learn the usage of various words and how the language is written in general. The model is now transferred to an NLP task where it is fed another smaller task-specific dataset, which is used to fine tune and create the final model capable of performing the aforementioned task.[7]

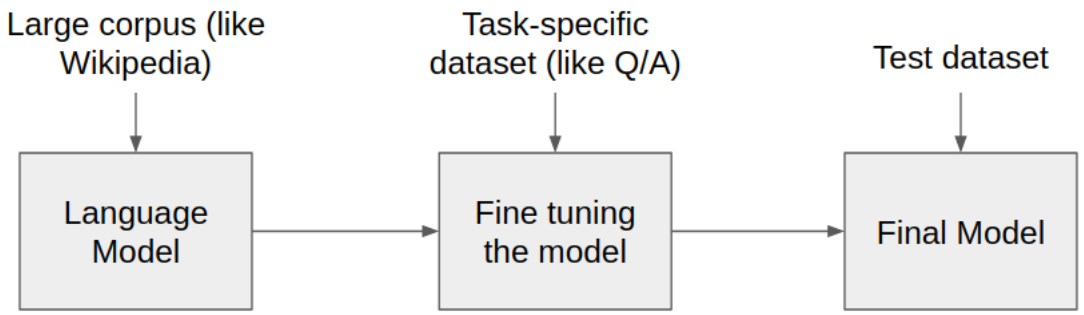


Figure 4: Language Model Pre-Training Workflow.[7]

Language model pre-training are extremely effective for improving many natural language processing task. As the *ULMfit* fine-tuning approach of Language-Model pre-training has showed.[8]

2.2.2 BERT Model

BERT, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers is designed to pre-train deep bidirectional representations from unlabeled text by jointly condition on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of NLP tasks.[9]

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing GLUE[10] score to 80.5% , MultiNLI[11] accuracy to 86.7%, SQuAD v1.1[12] question answering test F1 to 93.2 and SQuAD v2.0[12] Test F1 to 83.1.[9]

2.3 Related Work

IIT Bombay did some research on developing strategies for Sentiment Analysis in Hindi. In the paper they considered three approaches. The first to construct a classifier model for Hindi using a training corpus in Hindi. The second approach is to train a model on annotated English corpus and translate a Hindi document to English in order to use this model. The third approach involves using a majority-based classifier for Hindi SentiWordNet. The results of the research show that the first approach outperforms the others.[13]

In the paper titled BHAAV - A Text Corpus for emotion analysis from Hindi Stories, authors introduce the first and largest Hindi text corpus named BHAAV. The corpus consists of 20,304 sentences collected from 230 different short stores spanning across 18 genres. Each sentence has been annotated into one of the five emotion categories $\{$ *anger, joy, suspense, sad, and neutral*$\}$. The paper also discuss challenges in the annotation of low resource language such as Hindi, and discuss the proposed corpus along with its possible uses.[14] My first try for this project was to source this corpus for training our model, but since we couldn't get permission for the dataset we decided to use the approach presented in the next chapter.

3 Methodology and Implementation

This section details the methodology and implementation details of the project. In this section the following details are described - features of the dataset, preprocessing, and overall preparation, model topology, training and testing methodology.

3.1 Dataset

After the initial failure to get the access to the BHAAV dataset[14] , it was decide to find other resources for dataset. As Hindi is a low resource language this seemed to be a very difficult task at hand. Finally, we found a github repository[15] which had the dataset appropriate for sentiment analysis. The dataset here contains various tweets in Hindi along with corresponding label describing the emotional tone of the tweet. The categories of the label are *negative*, *neutral*, *positive*. Figure 5 shows some record in the dataset.

	column_0	column_1
0	लोग वतन तक खा जाते हैं इसका इसे यकीन नहींमान ज...	negative
1	गुमनाम है वतन पर मिटने वाले लोग आतन्कवादियों स...	negative
2	ज़ंजीर बदली जा रही थी मैं समझा था रिहाई हो गयी है	negative
3	यूपी में बड़े स्तर पर दंगे करवा सकती है बीजेपी...	negative
4	अंग्रेजी नहीं आती है इसलिए हिन्दी ट्विट ज्यादा...	negative

Figure 5: Subset of the dataset used in training the model.

The dataset contains ***9077*** number of record. This dataset is then further divided into training, test and validation set. The total records in training set are ***6353*** and the validation and test sets contains ***1362*** and ***1362*** records respectively. As, we can see ***6353*** is comparatively a small number of records required to give a good accuracy on task like Sentiment Analysis. This is where transfer learning comes into play.

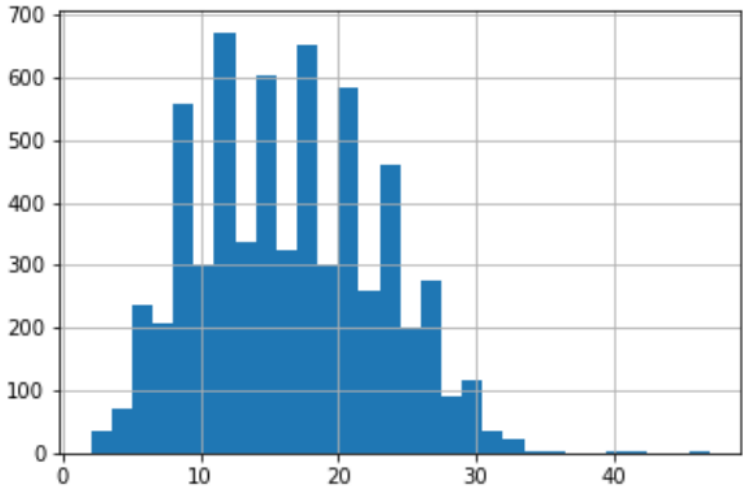
3.2 Preprocessing

There is not a lot of preprocessing done to the dataset. The first thing that is done is to encode the categorical data in label column as the model works on numerical values not text values. Label Encoding technique is used for this purpose. **Label Encoding** refers to converting the labels into numeric form so as to convert it into the machine-readable form. Machine learning algorithms can then decide in a better way on how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.[16] We have three categories in our label data namely *negative*, *neutral*, and *positive*. These will be converted to *0* , *1* and *2* respectively.

Secondly, for input we need to tokenize input as most NLP model are trained on tokens. For this we need to first make all the inputs of number of words . We do this by finding a cutoff number of words. For this we draw a histogram plot of all the inputs based on the total number of words they have and choose number of words that would satisfy most of the inputs. Figure 6 show the histogram distribution of input based on the total no. of words present in the piece of writing. And as we can see 35, is a good place to put our cutoff. So that's what we've done.

```
1 seq_len = [len(i.split()) for i in x_train[:,0]]
2 pd.Series(seq_len).hist(bins = 30)
```

```
<AxesSubplot:>
```




```
1 max_seq_len = 35
```

Figure 6: Histogram plot of the inputs based on the number of words

.

Then we have used to BERTTokenizerFast , a tokenizer pre-trained on multilingual dataset corpus. Tokenizer prepares our input in the following ways.

- Tokenizing (splitting strings in sub-word token strings), converting tokens strings to ids and back, and encoding/decoding (i.e., tokenizing and converting to integers).[17]
- Adding new tokens to the vocabulary in a way that is independent of the underlying structure (BPE, SentencePiece...).[17]
- Managing special tokens (like mask, beginning-of-sentence, etc.): adding them, assigning them to attributes in the tokenizer for easy access and making sure they are not split during tokenization.[17]

After the tokenization process is done we've feed these tokens as pytorch[18] tensors to be later transformed into pytorch DataClasses, which will be then used for training and validating in batches of fixed size. We have selected the batch size of 32.

3.3 Model

Our model has the following topology.

- First our model has the BERT pre-trained multilingual model. To which the preprocessed input is passed to.
- Then the output of the BERT model is fed to the first layer of our model which contains 768 input, and 512 output Nodes. We have used the **Re**ctified **L**inear **U**nit activation function also called ReLU function at this layer. It is defined as $R(z) = max(0,z)$. It is usually a good first choice for an activation function based on some empirical evidences.
- Then we have used a dropout layer to which the output from first layer is passed to. This layer prevents from the problem of over fitting which is a very common issue in training models on complex datasets. The dropout layer helps us fix this issue by ignoring some of the input nodes. [19]
- The output from the dropout layer is passed to our output layer. The output layer consists of 512 input nodes (same as the output of first layer) and 3 output nodes. As we have three categories to classify our data, therefore we've used 3 output

nodes. The activation function used in this layer is the softmax function. The standard softmax function: $\sigma : \mathbb{R}^K \to [0,1]^K$ is defined as $\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ $for\ i = 1,...,K\ and\ z = (z1,...,z_K) \in \mathbb{R}^K$. The softmax function is used to normalize the output of the network to a probability distribution over predicted output classes. [20]

We use *sklearn.utils.class_weight.compute_class_weight* library function to estimate initial weights for our model. This function estimates weights using heuristic inspired by Logistic Regression in Rare Events Data, King, Zen, 2001.[21] To calculate the loss we are using the cross entropy loss function. For a multiclass classification problem as we have, the cross entropy loss function can be defined as $-\sum_{c=1}^{M} y_{o,c} \ln(p_o, c)$.[22]

3.4 Training and Validation Process

Before training starts, we want to freeze the Bert Layers of our model so that the back propagation doesn't upgrade its weight. This is because we have a relatively small dataset and this can cause overfitting. So to avoid it, we freeze all the Bert Layers of our model. Then we decide on number of epochs we want to train our model for. For this project we have choose a relatively small epoch number which is ***100***. Now, our testing and validation process follow the given work flow.

- We step through the dataset in batches (batch size **32**).
- We send the preprocessed inputs from the batches to our model and get our predictions.
- We calculate the cross entropy loss from predicted and desired output.
- Then we update the weights of our Model using back-propagation.
- Then we perform the validation of our model weights. The process of validation is similar to the process of training but in validation we calculate the average loss for each epoch and doesn't upgrade the weights of the model. Model weights are only upgraded when training.
- We also save the weights of the best Model to a file in validation step. We decide which model is better using average loss that is calculated. We perform this check in every epoch.

This technique ensures that the model that is performing best in validation stage is the only one selected. This helps us removing overfitting even more as the model has no idea about the validation dataset during the training phase. Figure 7 showcase the basic idea behind training. Our training and validation loss decrease after each epoch(**this is not completely achievable all the time**.)

```
 Epoch 1 / 100
  Batch    50  of    199.
  Batch   100  of    199.
  Batch   150  of    199.

Evaluating...

Training Loss: 0.870
Validation Loss: 0.743

 Epoch 2 / 100
  Batch    50  of    199.
  Batch   100  of    199.
  Batch   150  of    199.

Evaluating...

Training Loss: 0.786
Validation Loss: 0.713
```

Figure 7: Our programs output showcasing the best validation and test loss for each epoch.

3.5 Testing Process

After the training is complete, the weights of the model with the best validation score (least average loss in validation phase) will be saved in a file. We load these weights to a new instance of our model, then we use this model to generate prediction of our test set.

The predictions generated from our model is then used along with the labeled data of the test dataset to generate a classification report detailing the performance of our model. The classification report along with its analysis is given in the next chapter

4 Result, Analysis and Conclusion

4.1 Result and Analysis

We've used the *sklearn.metrics.classification_report* method to generate the classification report of our model's performance. Table 2 shows the generated report in tabular form.

	precision	recall	f1-score	support
0	0.68	0.69	0.68	464
1	0.78	0.90	0.83	388
2	0.72	0.63	0.68	510
accuracy			0.73	1362
macro avg	0.73	0.74	0.73	1362
weighted avg	0.72	0.73	0.72	1362

Table 2: The classification report of our model's performance.

The aforementioned table consists of many classification metrics for all the three classes, i.e **0 - negative** , **1 - neutral**, and **2 - positive**, along with the overall accuracy of the model. The metrics as described in the scikit's Yellowbrick library are:

- **precision**
  Precision can be seen as a measure of a classifier's exactness. For each class, it is defined as the ratio of true positives to the sum of true and false positives. Said another way, "for all instances classified positive, what percent was correct?"

- **recall**
  Recall is a measure of the classifier's completeness; the ability of a classifier to correctly find all positive instances. For each class, it is defined as the ratio of true positives to the sum of true positives and false negatives. Said another way, "for all instances that were actually positive, what percent was classified correctly?"

- **f1 score**
  The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. Generally speaking, F1 scores are lower than accuracy measures as they embed precision and recall into their computation. As

a rule of thumb, the weighted average of F1 should be used to compare classifier models, not global accuracy.

- **support** Support is the number of actual occurrences of the class in the specified dataset. Imbalanced support in the training data may indicate structural weaknesses in the reported scores of the classifier and could indicate the need for stratified sampling or re balancing. Support doesn't change between models but instead diagnoses the evaluation process.

The **macro average** can be described as the average of the metric for all the classes *without* taking the proportion of each class into account. The **weighted average** is the average of the metric for all the classes *with* taking the proportion of each label into account.

4.2 Conclusion

So, with the overall accuracy of **0.73** or **73%** we can say that this is a decently good result for a relatively small dataset. This showcase the promise of transfer learning in the domain of Natural Language Processing for low resource languages like **Hindi**. Our classification report also shows that we performed very good for **neutral** cases as compared to the other too. We believe this is due to not given enough time and resource to train, the model doesn't get to learn polarity much adequately but nothing can be said with certainty at this point. In the next chapter, we highlight some of the future work and improvements that will increase the robustness and performance of our model.

5 Further work and improvement

The model and the process detailed in this dissertation can be improved with more work.

- The dataset contains lots of data that is not required for our desired working, like user-names and hash-tags. We can further preprocess the dataset to remove these undesired components. This may increase the performance of our model.
- Currently we have tested our model for only one value of max length, we can choose a better max length value through the process of cross validation. Choosing a different max length may increase the overall accuracy of our model.
- The topology of our model was restricted by the available VRAM of the machine. We would like to create either a more dense network or add more layers to the first and output layers our model. This may or may-not affect the overall performance of our model but as NLP models generally perform very well with mode dense networks. This change is likely to help.
- We could also train the model for more epochs. As 100 epochs may not be sufficient for the task at hand. We chose 100 epochs as it allowed us to train our model in decent time-frame.
- We could also do a cross analysis of this approach to a different approach of Sentiment Analysis using Transfer Learning.

With the addition of aforementioned works we may have a much more robust and precise model at hand. We could also try to replicate this approach on either a bigger corpus and differently classed corpus like that available in BHAAV dataset. [14]. These are some of the improvements and further works we suggest for our dissertation project.

Next Chapter contains all the References used in the making of this project.

References

[1] (May 30, 2019). "Guide to transfer learning with real-world applications in deep learning," ThirdEye Data. Section: Deep Learning, [Online]. Available: https://thirdeyedata.io/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning/ (visited on 07/27/2021).

[2] D. D. Sarkar. (Dec. 13, 2018). "Deep transfer learning for natural language processing — text classification with universal...," [Online]. Available: https://towardsdatascience.com/deep-transfer-learning-for-natural-language-processing-text-classification-with-universal-1a2c69e5baa9 (visited on 07/25/2021).

[3] S. Smetanin, "The applications of sentiment analysis for russian language texts: Current challenges and future perspectives," *IEEE Access*, vol. 8, pp. 110 693–110 719, 2020, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3002215.

[4] (Mar. 21, 2017). "Transfer learning - machine learning's next frontier," Sebastian Ruder, [Online]. Available: https://ruder.io/transfer-learning/ (visited on 07/27/2021).

[5] Y.-P. Lin and T.-P. Jung, "Improving EEG-based emotion classification using conditional transfer learning," *Frontiers in Human Neuroscience*, vol. 11, p. 334, Jun. 27, 2017, ISSN: 1662-5161. DOI: 10.3389/fnhum.2017.00334. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5486154/ (visited on 07/27/2021).

[6] R. Liu, Y. SHI, C. JI, and M. JIA, "A survey of sentiment analysis based on transfer learning," *IEEE Access*, vol. PP, pp. 1–1, Jun. 26, 2019. DOI: 10.1109/ACCESS.2019.2925059.

[7] P. Ganesh. (Dec. 17, 2019). "Pre-trained language models : Simplified," Medium, [Online]. Available: https://towardsdatascience.com/pre-trained-language-models-simplified-b8ec80c62217 (visited on 07/27/2021).

[8] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," *arXiv:1801.06146 [cs, stat]*, May 23, 2018. arXiv: 1801.06146. [Online]. Available: http://arxiv.org/abs/1801.06146 (visited on 07/27/2021).

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *arXiv:1810.04805 [cs]*,

May 24, 2019. arXiv: 1810.04805. [Online]. Available: http://arxiv.org/abs/1810.04805 (visited on 07/27/2021).

[10] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman, "GLUE: A multi-task benchmark and analysis platform for natural language understanding," in *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 353–355. DOI: 10.18653/v1/W18-5446. [Online]. Available: http://aclweb.org/anthology/W18-5446 (visited on 07/27/2021).

[11] A. Williams, N. Nangia, and S. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana: Association for Computational Linguistics, 2018, pp. 1112–1122. DOI: 10.18653/v1/N18-1101. [Online]. Available: http://aclweb.org/anthology/N18-1101 (visited on 07/27/2021).

[12] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, "SQuAD: 100,000+ questions for machine comprehension of text," *arXiv:1606.05250 [cs]*, Oct. 10, 2016. arXiv: 1606.05250. [Online]. Available: http://arxiv.org/abs/1606.05250 (visited on 07/27/2021).

[13] A. Joshi, "A fall-back strategy for sentiment analysis in hindi: A case study," p. 6,

[14] Y. Kumar, D. Mahata, S. Aggarwal, A. Chugh, R. Maheshwari, and R. R. Shah, *BHAAV - a text corpus for emotion analysis from hindi stories*, Type: dataset, Sep. 22, 2019. DOI: 10.5281/ZENODO.3457467. [Online]. Available: https://zenodo.org/record/3457467 (visited on 07/28/2021).

[15] S. Sinha, *Sid573/hindi_sentiment_analysis*, original-date: 2019-06-11T09:27:27Z, May 13, 2021. [Online]. Available: https://github.com/sid573/Hindi_Sentiment_Analysis (visited on 07/28/2021).

[16] (Oct. 15, 2018). "ML | label encoding of datasets in python," GeeksforGeeks. Section: Machine Learning, [Online]. Available: https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/ (visited on 07/28/2021).

[17] (). "Tokenizer," [Online]. Available: https://huggingface.co/transformers/main_classes/main_classes/tokenizer.html (visited on 07/28/2021).

[18] (). "PyTorch," [Online]. Available: https://www.pytorch.org (visited on 07/28/2021).

[19] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," p. 30,

[20] *Softmax function*, in *Wikipedia*, Page Version ID: 1030172347, Jun. 24, 2021. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Softmax_function&oldid=1030172347 (visited on 07/28/2021).

[21] G. King and L. Zeng, "Logistic regression in rare events data," p. 27, 2001.

[22] (). "Loss functions — ML glossary documentation," [Online]. Available: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html (visited on 07/28/2021).