HPDC – 438: Team 16 Project Report

Analyzing Amazon Product Reviews: Insights into Sentiment Analysis Of Product Reviews And Product Ratings

Academic Year: Fall 2023

Authors:
Ajinkya Pophale (axp1079)
Sai Dheeraj Yanduru (sxy874)
Manish SSS Routhu (mxr809)
Amulya Pophale (axp1142)
Akhil Reddy Sheri (axs2613)
Veerendra Varma Mavulate (vxm262)

TABLE OF CONTENTS

Introduction 2

Project timelines 3

Project Workflow 7

Data sampling and initial Data Exploration 7

Initial Data Visualization and EDA 8

HPC setup 9

Sentiment Analysis / Data Pre-processing 11

Models used / feature engineering 13

Recurrent Neural Networks 13

LSTM (Long Short-Term Memory) 15

Bi LSTM: Bidirectional Long Short-Term Memory 16

BERT -Bidirectional Encoder Representations from Transformers 20

Results and conclusion 21

References 23

INTRODUCTION

In an era of unprecedented e-commerce growth, the analysis of user-generated product reviews and ratings has become paramount for businesses seeking to understand customer sentiment and enhance product offerings. This project embarks on a colossal journey, undertaking the analysis of an extensive dataset comprising nearly 200,000 Amazon product reviews.

Our primary focus is to extract profound insights utilizing sentiment analysis and to unveil trends in product ratings. The project leverages advanced machine learning algorithms and harnesses the formidable power of a high-performance computing (HPC) cluster to enable efficient processing of this vast data repository.

The project commences with data collection and pre-processing, addressing challenges posed by the scale and diversity of Amazon product reviews. With a robust and cleanse dataset in hand, we delve into sentiment analysis, utilizing natural language processing techniques and machine learning models to discern the emotional tone of customer reviews. Simultaneously, we delve into the realm of product rating analysis, scrutinizing the distribution of ratings and product performance trends over time.
High-performance computing clusters play a pivotal role in the project, enabling the efficient analysis and processing of a dataset of this magnitude. Distributed computing frameworks are employed to parallelize computations, ensuring timely and scalable analysis.

The results of this endeavour encompass a comprehensive understanding of customer sentiment, patterns in product ratings, and the exploration of factors influencing customer satisfaction.
The project's findings have potential implications for businesses and product developers, offering actionable insights for decision-making and product improvements. Furthermore, the methodologies and techniques employed provide a template for handling vast datasets and leveraging high-performance computing resources for data analysis.
In conclusion, this project bridges the gap between colossal datasets, sophisticated machine learning techniques, and high-performance computing infrastructure to uncover valuable insights in the realm of e-commerce and consumer sentiment analysis.

PROJECT TIMELINES

Achieving significant results in a data analysis project, especially one as extensive as "Analysing Amazon Product Reviews," within a three-week time frame is challenging. However, with careful planning and efficient use of resources, we made substantial progress and accomplished certain key tasks. Here's what we were able to achieve in three weeks:

1. Data Preparation (Week 1):

- Data Collection: Acquired a subset of the dataset for initial analysis.
- Ensured that we have all the necessary files and data formats, including reviews, product information, and user details.
- Data Pre-processing: Clean, format, and structure the data for analysis. This includes handling missing values, deduplication, and basic data cleaning.
- Data Transformation:
  - Transformed the data to make it more suitable for analysis. Common transformation tasks include:
    - Feature Engineering: Created new features/variables that are relevant for our analysis. For example, we extracted sentiment features from the review text.
    - Text Pre-processing: Applied text pre-processing techniques like tokenization, stemming, and removing stop words to the review text.
    - Aggregation: Summarized data at different levels, such as aggregating reviews by product, category, or time.

2. Initial Data Exploration and Sampling (Week 1):

- Performed an initial data exploration to understand the dataset's characteristics, such as the distribution of reviews and product categories.
- Randomly sampled a portion of the data to work with, allowing for faster processing and modelling.

3. Sentiment Analysis (Week 2):

- Followed sentiment analysis approach and set up the necessary libraries and tools.
- Trained a preliminary sentiment analysis model on the sampled data. It is not as accurate as a fully trained model, but it provides a general idea of sentiment trends.
- Evaluated the model's performance on the sample.

4. Data Visualization (Week 2):

- Created initial data visualizations to present insights from the sampled dataset. These visualizations can provide a high-level overview of the data.

5. Product Rating Analysis (Week 2):

- Performed basic statistics, such as the mean and distribution of product ratings, using the sampled data.
- Visualized the distribution of ratings and trends.

6. High-Performance Computing Setup (Week 2):
- We set up the HPC cluster and ensured its operational for further processing.

In our project, we've reached a stage where we've prepared the data and conducted preliminary analysis on a subset of the Amazon product review dataset.

1. **Cluster Configuration**:
    - Configured an HPC cluster with a sufficient number of compute nodes. The cluster's architecture matches the requirements of our data analysis tasks.
2. **Parallelization**:
    - Implemented parallel processing for our sentiment analysis and product rating analysis on the cluster. This enabled us to distribute the computational load across multiple nodes, significantly improving processing speed and efficiency.

7. Documentation and Reporting (Week 3):
- Began documenting our work, methods, and preliminary findings.
- Created a report or presentation outline for our final project documentation.

8. Optimization (Week 3):
- Identified areas where we can optimize our process and consider future steps for more in-depth analysis.

To complete a project of the scale and complexity of "Analysing Amazon Product Reviews: Insights into Sentiment Analysis of Product Reviews and Product Ratings using ML Algorithms and High-Performance Computing," we'll need to continue with various tasks in the following weeks. We have completed the initial data preparation and preliminary analysis, here's what we did in the next stages of the project:

Week 4:

1. Full-Scale Data Analysis:
- Scale up our sentiment analysis and product rating analysis to work with as much data as our available resources and time permits. This involves processing a larger sample or a partition of the entire dataset.

2. Advanced Sentiment Analysis:
- Fine-tune our sentiment analysis model for improved accuracy and robustness. Consider experimenting with different machine learning algorithms or pre-trained language models for better results.

3. Data Visualization and Insights:
- Create more comprehensive data visualizations and derive deeper insights from the larger dataset. Explore trends in sentiment over time, by product category, or other relevant factors.

Week 5:

5. Feature Engineering:
- Expand our feature engineering efforts to capture additional relevant information from the text data. This can enhance the accuracy of our sentiment analysis and provide more context for product ratings.

6. Model Evaluation:
- Rigorously evaluate the performance of our sentiment analysis and product rating models. Use of appropriate metrics, cross-validation, and testing to ensure their reliability.

7. Results Documentation:
- Document our results, findings, and insights in a structured manner. Create visualizations and tables to represent our analysis comprehensively.

Week 6:

8. Final Report/Presentation:
- Compile our results and insights into a final report or presentation. Include details about our methodology, findings, and any recommendations for businesses or future research.

9. Future Directions:
- Discuss possible future directions for the project, such as additional analysis, improvements in sentiment analysis, or expanding the scope to include other e-commerce platforms.

10. Review and Refinement:
- Review our entire project, seek feedback from peers or mentors, and refine any areas that need improvement. Ensure our documentation is clear and comprehensive.

11. Project Wrap-Up:
- Conclude the project by organizing all project artifacts, source code, and documentation. Ensure that our analysis can be easily reproduced by others.

12. Presentation and Delivery:
- Present our findings and insights to the intended audience, whether it's a class, research group, or stakeholders. Prepare to answer questions and engage in discussions.

Team Responsibilities:

Here's a division of responsibilities among our team members for the project "Analysing Amazon Product Reviews." Few of the roles and responsibilities overlap as teamwork is essential. This division is based on common tasks within our project:

1. Project Management:
- *All team members collectively:*
- Responsible for overall project management, communication, and coordination among team members.
- Ensures that project milestones and deadlines are met.

- Facilitates regular team meetings and serves as the main point of contact.

2. Data Acquisition and Preprocessing:
- *Ajinkya Pophale, Sai Dheeraj Yanduru, and Amulya Pophale:*

- Responsible for obtaining and preprocessing the raw Amazon product review dataset.
- Ensure data quality and cleanliness, including data cleaning, handling missing values, and deduplication.

3. Sentiment Analysis and ML Modelling:
- *Akhil Reddy Sheri, Veerendra M, and Manish Routhu:*

- Lead the development and fine-tuning of sentiment analysis models.
- Implement machine learning algorithms for sentiment analysis.
- Evaluate and validate the model's performance.

4. Product Rating Analysis:
- *Sai Dheeraj Yanduru, Amulya Pophale, and Veerendra M:*

- Lead the analysis of product ratings, including calculating statistics and visualizing rating distributions.
- Explore trends in product ratings over time or by category.

5. High-Performance Computing (HPC) and Cluster Setup:
- *Sai Dheeraj Yanduru, Ajinkya Pophale, and Veerendra M:*

- Collaborate on setting up and configuring the HPC cluster for data processing.
- Implement parallel processing for data analysis tasks on the cluster.
- Monitor and optimize cluster performance as needed.

6. Data Visualization and Reporting:
- *Akhil Reddy Sheri, Amulya Pophale, and Manish Routhu:*

- Create data visualizations to represent findings and insights.
- Collaborate on generating the project's final report or presentation.

7. Documentation and Code Repository:
- *Akhil Reddy Sheri, Ajinkya Pophale, and Manish Routhu:*

- Ensure that all project activities, code, and results are well-documented.
- Maintain a shared code repository for version control and collaboration.

8. Project Presentation and Communication:
- *All team members collectively:*

- Collaborate on preparing and delivering project presentations to the intended audience.
- Maintain effective communication within the team and with project stakeholders.

This division of responsibilities is a starting point and could be adjusted based on the skills and preferences of each team member. Effective communication and collaboration are key to the success of our project, so regularly we update each other on progress and challenges, and prepare to adapt roles as needed to meet project goals and deadlines.

PROJECT WORKFLOW

The project commences with data collection and pre-processing, addressing challenges posed by the scale and diversity of Amazon product reviews.

With a robust and cleansed dataset in hand, we delve into sentiment analysis, utilizing natural language processing techniques and machine learning models.

Simultaneously, we delve into the realm of product rating analysis, scrutinizing the distribution of ratings and product performance trends over time.

High-performance computing clusters play a pivotal role in the project, enabling the efficient analysis and processing of a dataset of this magnitude.

We utilized TensorFlow's mirrored strategy to enable distributed computing, allowing parallel processing for our analyses, and ensuring timely and scalable operations.

The results of this endeavour encompass a comprehensive understanding of customer sentiment, patterns in product ratings, and the exploration of factors influencing customer satisfaction.

DATA SAMPLING AND INITIAL DATA EXPLORATION

This Dataset is an updated version of the Amazon review dataset released in 2014. As in the previous version, this dataset includes reviews (ratings, text, helpfulness votes), product metadata (descriptions, category information, price, brand, and image features), and links (also viewed/also bought graphs). In addition, this version provides the following features:

The total number of reviews is 233.1 million (142.8 million in 2014). However, we will be using only 200k records for our project.

Metadata:

```
reviews.head()
```

	reviewerID	asin	reviewerName	helpful	reviewText	overall	summary	unixReviewTime	reviewTime
0	A30TL5EWN6DFXT	120401325X	christina	[0, 0]	They look good and stick good! I just don't li...	4.0	Looks Good	1400630400	05 21, 2014
1	ASY55RVNIL0UD	120401325X	emily l.	[0, 0]	These stickers work like the review says they ...	5.0	Really great product.	1389657600	01 14, 2014
2	A2TMXE2AFO7ONB	120401325X	Erica	[0, 0]	These are awesome and make my phone look so st...	5.0	LOVE LOVE LOVE	1403740800	06 26, 2014
3	AWJ0WZQYMYFQ4	120401325X	JM	[4, 4]	Item arrived in great time and was in perfect ...	4.0	Cute!	1382313600	10 21, 2013
4	ATX7CZYFXI1KW	120401325X	patrice m rogoza	[2, 3]	awesome! stays on, and looks great. can be use...	5.0	leopard home button sticker for iphone 4s	1359849600	02 3, 2013

Columns used in our data sample ex:

ReviewText: Product reviews which will be sent to the models to learn.

Overall: score of the product ratings corresponding to the reviews

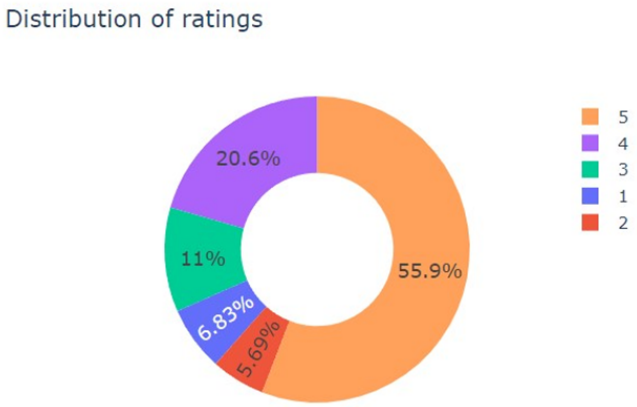
	reviewText	overall
194434	Works great just like my original one. I reall...	5.0
194435	Great product. Great packaging. High quality a...	5.0
194436	This is a great cable, just as good as the mor...	5.0
194437	I really like it becasue it works well with my...	5.0
194438	product as described, I have wasted a lot of m...	5.0

INITIAL DATA VISUALIZATION AND EDA

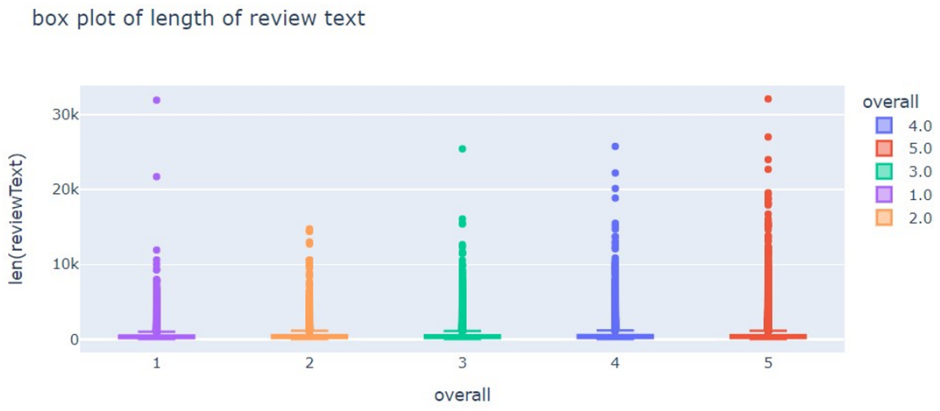
Data distribution of Overall column

Overall (Ratings)	Data points
1	13269
2	11059
3	21436
4	39974

5	108602



The length of the review vs rating for the product is also plotted which is shown below:



HPC SETUP

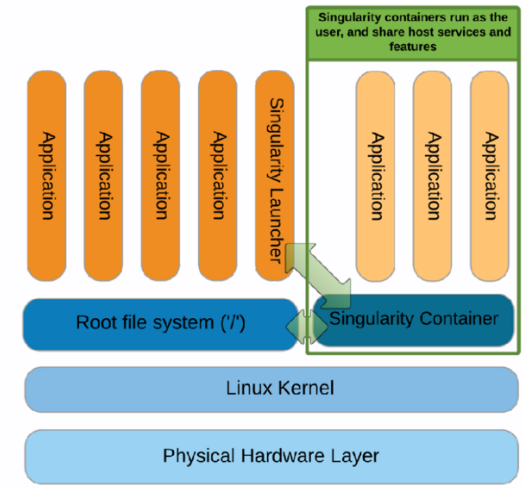
Introduction:

In the pursuit of optimizing our model training processes, we have adopted a streamlined approach utilizing the TensorFlow module in Python. The key focus has

been on leveraging the power of High-Performance Computing (HPC) resources to efficiently train our models in a distributed environment.

Methodology:

To facilitate the deployment of our TensorFlow models, we have embraced containerization using Docker and Singularity. This has allowed us to encapsulate our models and their dependencies within a suitable TensorFlow image, ensuring consistency and reproducibility across various computing environments.



HPC Infrastructure:

For the execution of our models, we secured a dedicated node equipped with two GPUs. This strategic choice was made to capitalize on the parallel processing capabilities of Graphics Processing Units, enhancing the speed and efficiency of our model training.

Distributed Training:

Our approach to model training involves the orchestration of distributed training across the two GPUs available on the allocated node. This methodology ensures that each model benefits from parallelization, optimizing computational resources and reducing training time.

Execution:

The process is executed uniformly across all models, ensuring a fair and consistent evaluation of performance. This approach not only enhances efficiency but also provides a comparative basis for assessing the effectiveness of each model under similar conditions.

Results and Considerations:

By implementing this methodology, we aim to achieve not only optimal training times but also to explore the scalability and performance of our models in a distributed environment. The use of Docker and Singularity containers ensures that our models are encapsulated and transportable, promoting seamless deployment across various computational architectures.

Conclusion:

In conclusion, our adoption of TensorFlow, Docker, and Singularity in conjunction with distributed training on an HPC cluster represents a comprehensive strategy to enhance the efficiency and scalability of our model training processes. This approach lays the foundation for future advancements in model development and training optimization.

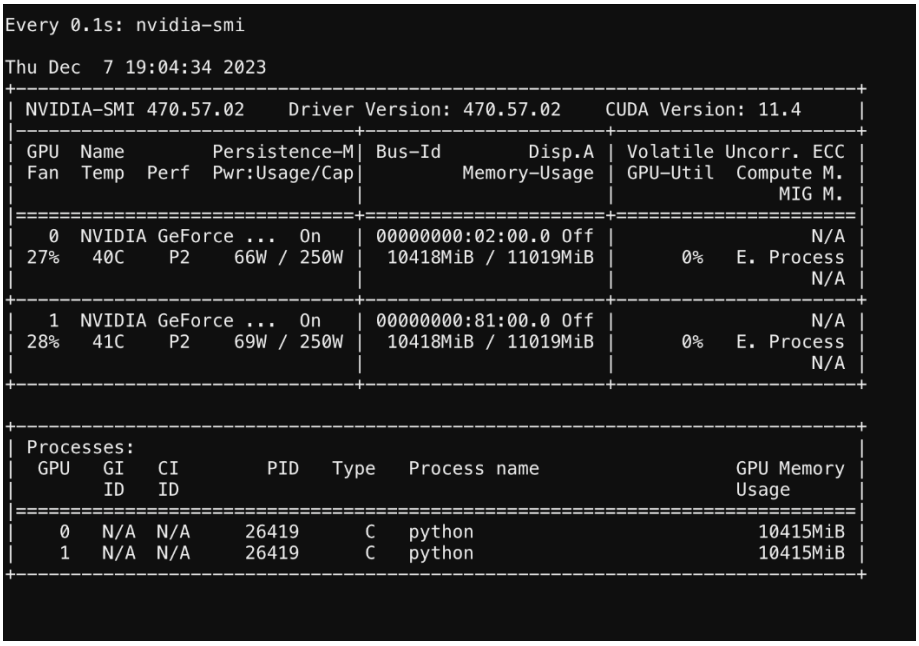


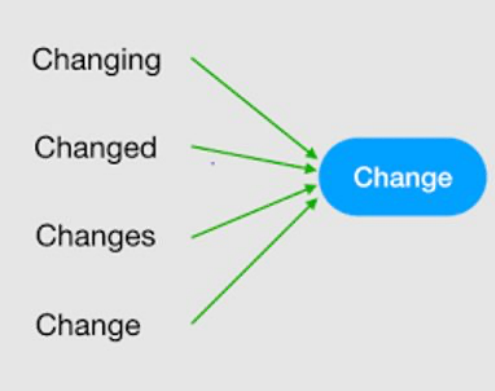
Fig: Snapshot of GPU usage while training our models on the HPC.

SENTIMENT ANALYSIS / DATA PRE-PROCESSING

Lemmatization:

Lemmatization is a linguistic process that involves reducing words to their base or root form, known as the lemma.
The goal of lemmatization is to group different inflected forms of a word, so they can be analysed as a single item.

This process is particularly important in natural language processing (NLP) and text analysis.

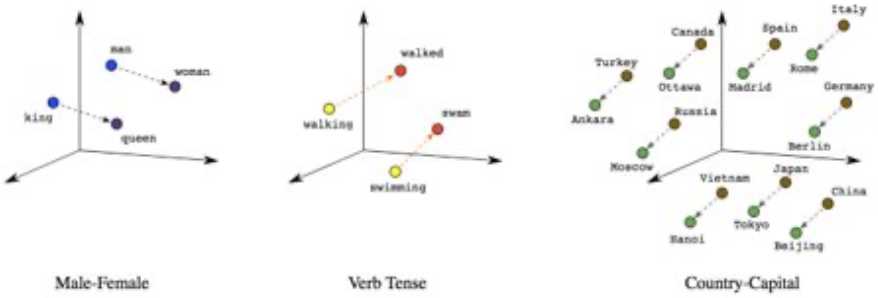


GloVe Embeddings:

GloVe, an unsupervised learning algorithm, derives word vectors by creating and factorizing a global word co-occurrence matrix from extensive text. These resulting embeddings encode semantic relationships and word similarities.

Average Embeddings:

Average embeddings create a document or sentence vector by averaging pre-trained word embeddings, providing a simpler and less computationally intensive method to capture the overall document or sentence meaning.



Male-Female Verb Tense Country-Capital

MODELS USED / FEATURE ENGINEERING

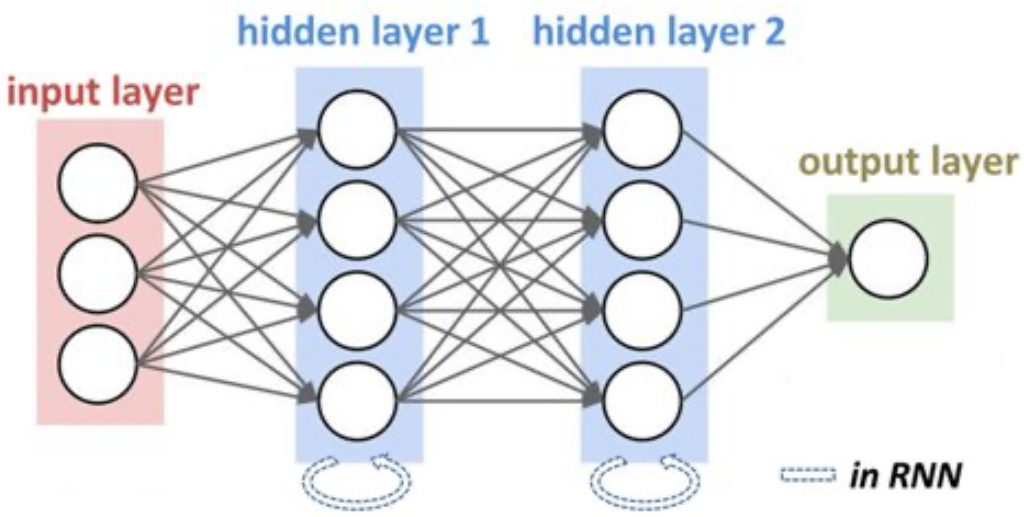
Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a class of neural networks designed to effectively handle sequential data by maintaining internal memory.

Sequential Modelling: RNNs process sequential data by considering the order and dependencies among elements in the sequence. Each element (or time step) in the sequence is processed one after another.

Internal Memory (Hidden State): RNNs maintain an internal memory (hidden state) that retains information about the previously seen elements in the sequence. This memory allows RNNs to capture context and dependencies across different time steps.

Shared Parameters: RNNs use the same set of weights and biases across all time steps, allowing them to learn patterns and relationships within sequences and generalize to sequences of varying lengths.



Sequential Model Initialization: This initializes a sequential model where layers are added sequentially.

Embedding Layer: This layer is an Embedding layer that maps input tokens (words represented as integers) to dense vectors of fixed size.

Recurrent Layers (SimpleRNN):

The SimpleRNN layers are added to the neural network model. These layers form the core of the model, processing sequential data while retaining memory across different time steps.

The inclusion of multiple SimpleRNN layers, combined with Dropout for regularization, aims to enhance the model's ability to learn and represent patterns within sequential data.

Last Hidden Layer:

This final SimpleRNN layer consolidates information learned from the entire sequence. A Dropout layer (20% dropout) follows the last SimpleRNN layer to prevent overfitting. This layer serves as the final representation of the sequential data before the output layer. It captures the most abstract and summarized information from the entire sequence to feed into the output layer for classification or prediction.

Output Layer:

This layer generates the final output by converting the learned features from the previous layers into class probabilities. Configured with a categorical cross-entropy loss function to measure the difference between predicted and actual class distributions during training.

Model Information: An RNN model with 4 layers and 100 neurons in each layer was used.

Accuracy:
Accuracy is a measure of the overall correctness of the model. It is calculated as the ratio of correctly predicted instances to the total instances.

$$Accuracy = \frac{Total\ Number\ of\ Predictions}{Number\ of\ correct\ Predictions}$$

F1 Score:
The F1 score is the harmonic mean of precision and recall. It is often used when there is an uneven class distribution (imbalanced classes).

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

$$F1\ Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

It's important to note that while accuracy is a good metric for balanced datasets, the F1 score might be more appropriate when dealing with imbalanced datasets, as it considers both false positives and false negatives. Depending on the context of your classification problem, you may choose the metric that best suits your evaluation needs.

Performance Metrics:	*F1 score*	*Accuracy score*
***Train Set*:**	0.07043	20.565%.
***Test Set*:**	0.07042	20.5668%.

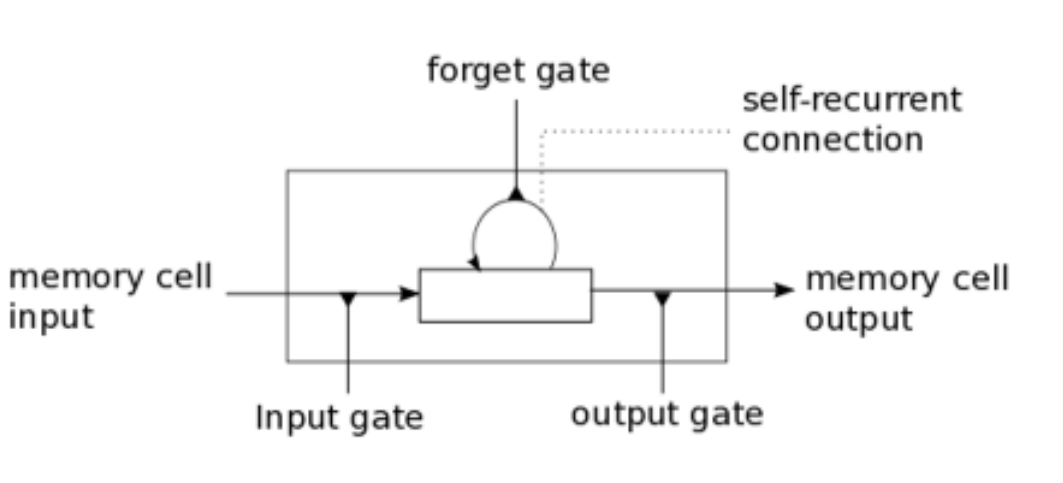
The reported F1 scores and accuracies indicate the model's performance on both the training and test sets. These values suggest a low performance level in terms of classification or prediction for the given task.
The model seems to struggle to capture patterns and make accurate predictions based on the provided data.

LSTM (Long Short-Term Memory)

LSTM (Long Short-Term Memory) is a type of recurrent neural network (RNN) designed to tackle the vanishing gradient problem in traditional RNNs and better capture long-range dependencies in sequential data.

Memory Cells: LSTMs use memory cells to store and regulate information flow over time. These cells maintain a memory state, allowing the network to retain information for longer durations.

Gates: LSTMs have three gates - input gate, forget gate and output gate. These gates control the flow of information, enabling the model to selectively add, remove, or output information to and from the memory cells.



Key components of LSTM:

Memory Cells: LSTMs use memory cells to store and regulate information flow over time. These cells maintain a memory state, allowing the network to retain information for longer durations.

Gates: LSTMs have three gates - input gate, forget gate and output gate. These gates control the flow of information, enabling the model to selectively add, remove, or output information to and from the memory cells.

Final hidden state: In the context of an RNN like an LSTM or a GRU (Gated Recurrent Unit), the final hidden state represents the network's understanding or summarization of the entire input sequence.

- We can make the units even more complex
- Allow each time step to modify
  - Input gate (current cell matters) $i_t = \sigma\left(W^{(i)}x_t + U^{(i)}h_{t-1}\right)$
  - Forget (gate 0, forget past) $f_t = \sigma\left(W^{(f)}x_t + U^{(f)}h_{t-1}\right)$
  - Output (how much cell is exposed) $o_t = \sigma\left(W^{(o)}x_t + U^{(o)}h_{t-1}\right)$
  - New memory cell $\tilde{c}_t = \tanh\left(W^{(c)}x_t + U^{(c)}h_{t-1}\right)$
- Final memory cell: $c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t$
- Final hidden state: $h_t = o_t \circ \tanh(c_t)$

Result

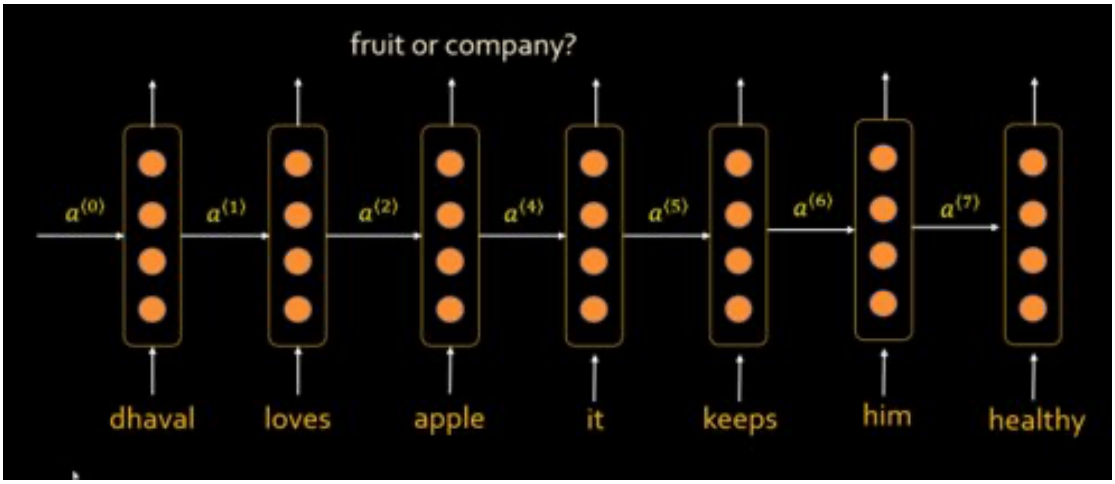
Performance Metrics:	*F1 score*	*Accuracy score*
***Train Set*:**	0.5585	0.53072
***Test Set*:**	0.5328	0.50123

We can observe that there is a significant improvement in the model performance compared to RNN.

This is because LSTMs can retain long-term dependencies in the data.

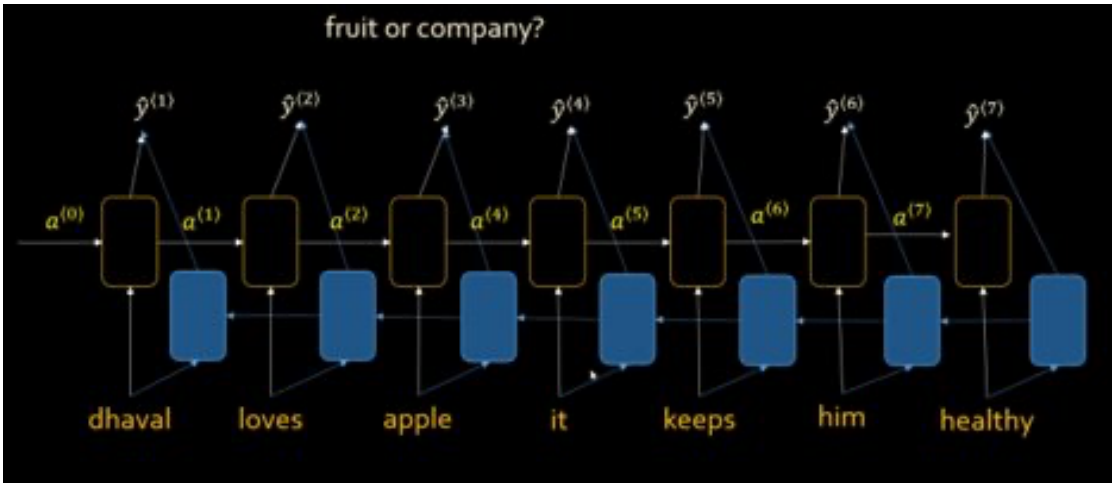
Bi LSTM: Bidirectional Long Short-Term Memory

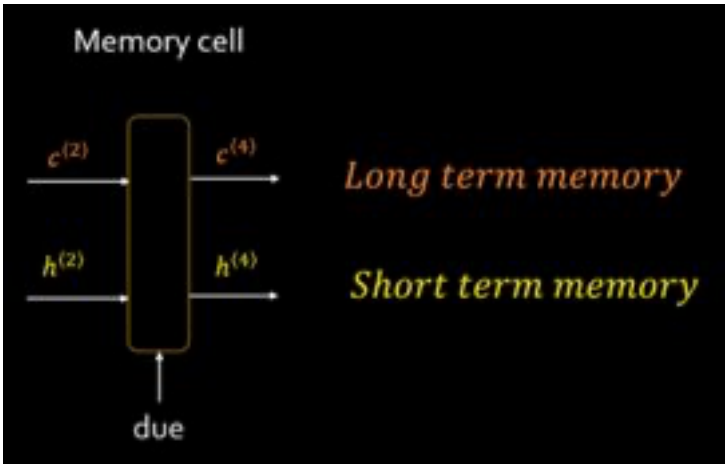
Introduction:
Transition into the need for addressing the bidirectional context in the sequential data processing



Limitations of Unidirectional Models:
Unidirectional models (like traditional LSTMs) process information in only one direction, lacking access to the future context.
such models might not fully comprehend the context in sequences where bidirectional information matters.

Example of Contextual Understanding:
Use a sentence or text snippet where understanding both preceding and succeeding words is crucial for accurate interpretation.
unidirectional models might struggle to grasp the complete meaning without bidirectional context.





Need for Bidirectional Models:
need to process both past and future contexts for a comprehensive understanding of sequential data. Emphasize that bidirectional models, like Bi LSTM, address this gap by capturing context from both directions.

Example: Understanding Ambiguous Sentences
Consider the sentence: "He saw the man on the hill with a telescope."

Unidirectional Model Interpretation: A unidirectional model might interpret this sentence solely based on its forward flow of information, leading to ambiguity. It may infer that "He" used the telescope while on the hill, but it doesn't consider alternative interpretations.

Bidirectional Context's Significance: Bidirectional context is crucial here. Understanding that the sentence could have multiple meanings:
With Forward Context (Unidirectional):
"He" saw someone "on the hill" and used a telescope.

With Bidirectional Context:
"He" saw someone. The other person was "on the hill" and used a telescope.

Interpretation Clarification: Bidirectional models like Bi LSTM, considering both the preceding and succeeding words, are better equipped to grasp the correct meaning by leveraging the entire context. This helps in disambiguating sentences and understanding nuanced meanings, showcasing the significance of bidirectional context in language comprehension.

- Dhaval loves apple, it keeps him healthy – here apple is the fruit
- Dhaval loves Apple, the company that produces the best electronics - here apple is a company

Bi LSTM faces several challenges compared to RNN and LSTM models

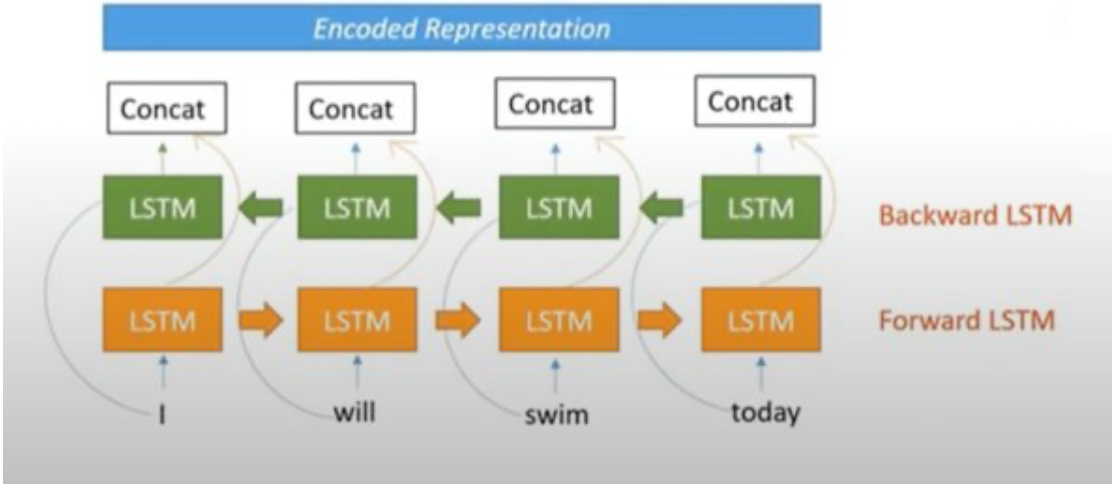
Language Modeling
One of the main challenges of using Bidirectional LSTM for language modelling is the issue of vanishing gradients. This occurs when the gradients of the model become very small as they propagate backwards in time, making it difficult to train the model effectively. To address this, techniques such as gradient clipping and gradient normalization have been proposed for this challenge.

Sentiment Analysis
Another challenge of using Bidirectional LSTM for sentiment analysis is the issue of overfitting. This occurs when the model becomes too complex and starts to memorize the training data, rather than learning generalizable patterns. To address this, techniques such as regularization and early stopping have been proposed.

Hyperparameter Tuning
One of the main challenges of Bidirectional LSTM is hyperparameter tuning. This involves finding the optimal values for parameters such as learning rate, batch size, and number of epochs. By using techniques such as grid search and random search, it is possible to find the best combination of hyperparameters for a given task.

Learning rate, batch size and number of epochs
In addition to these hyperparameters, several other parameters can be tuned for Bidirectional LSTM models, such as the number of layers, the type of activation function, and the type of recurrent neural network unit.

Model Architecture Modifications
Another approach to overcoming the challenges of Bidirectional LSTM is to modify the model architecture. This can involve adding or removing layers, changing the activation functions, or using different types of recurrent neural networks. By experimenting with different architectures, it is possible to find a configuration that performs well on a given task.



Results

Performance Metrics:	*F1 score*	*Accuracy score*
***Train Set*:**	0.5339	0.50321
***Test Set*:**	0.4919	0.45160

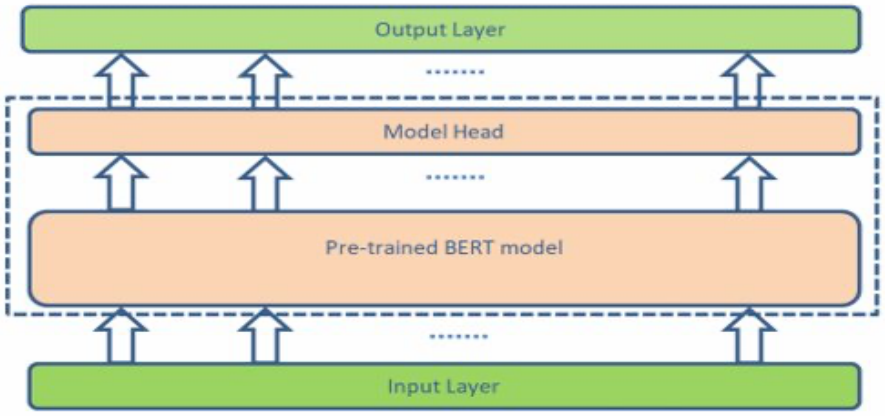
It's open-source libraries and a pre-trained model that has significantly advanced the field of NLP.

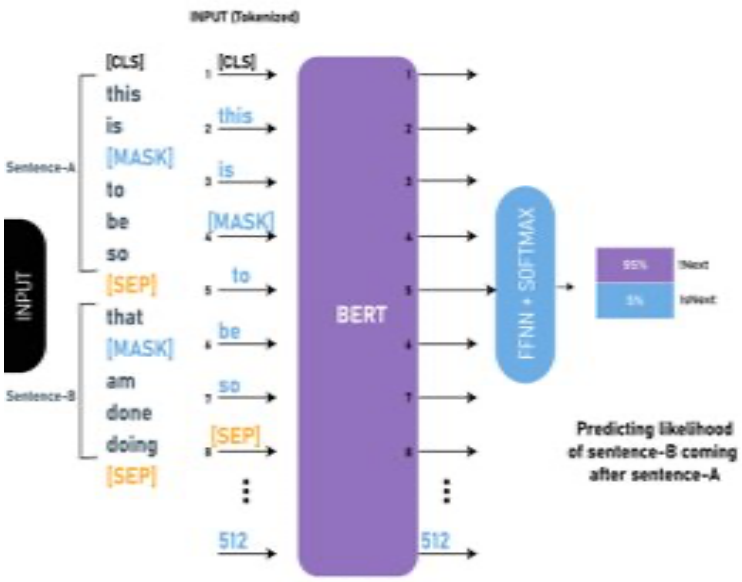
Transformers Library: Hugging Face developed the "Transformers" library, a popular open-source library that provides an easy-to-use interface for working with state-of-the-art transformer-based models in NLP. It offers a wide range of pre-trained models (such as BERT, and GPT) and tools for fine-tuning, training, and using these models for various NLP tasks.

BERT -Bidirectional Encoder Representations from Transformers

BERT is a natural language processing (NLP) model introduced by Google in 2018. looks at the entire sentence context in both directions during training. it is based on the Transformer architecture, which uses self-attention mechanisms to weigh the importance of different words in a sentence, allowing the model to consider the entire context when making predictions.

BERT has been pre-trained on large corpora of text data and has achieved state-of-the-art results on various natural language processing tasks, such as question answering, sentiment analysis, and language translation.





Result

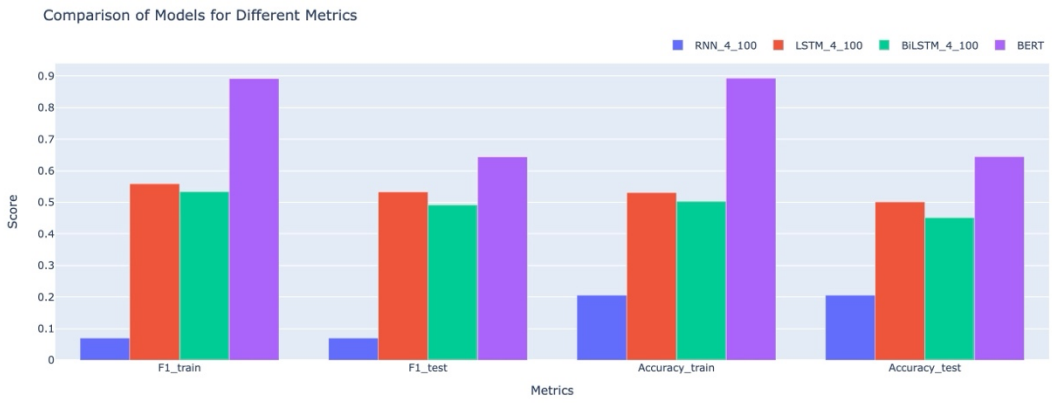
Performance Metrics:	*F1 score*	*Accuracy score*
***Train Set*:**	0.89183	0.893122
***Test Set*:**	0.64399	0.644543

As we can see we get the best results from BERT

RESULTS AND CONCLUSION

Model performances based on Performance metrics F1 and Accuracy

	F1 Train	*F1 Test*	*Accuracy Train*	*Accuracy Test*
RNN	0.070	0.074	0.205	0.256
LSTM	0.558	0.532	0.534	0.502
Bi LSTM	0.533	0.491	0.503	0.453
BERT	0.891	0.643	0.893	0.644



The project begins by tackling the significant task of collecting and preprocessing data from Amazon product reviews. This step involves gathering reviews from various sources, dealing with the vast scale and diversity inherent in such datasets. Preprocessing involves cleaning and organizing the data, handling challenges like noise, inconsistencies, and different formats to ensure the dataset's quality and usability.

Once a robust and refined dataset is prepared, the focus shifts to sentiment analysis. Natural language processing (NLP) techniques, coupled with machine learning models, are employed to analyze the text content of the reviews. Sentiment analysis aims to understand the sentiment expressed in the reviews, identifying positive, negative, or neutral sentiments associated with different products or aspects.

Simultaneously, the project delves into the realm of product rating analysis. This involves studying the distribution of ratings given by customers to products over time. Analyzing trends in ratings can reveal patterns, identify shifts in consumer perception, and highlight factors impacting product performance.

A critical aspect enabling the project's analysis is the utilization of high-performance computing clusters. These clusters offer the necessary computational power to efficiently handle and process the vast amounts of data involved in this project.

To optimize computational efficiency further, TensorFlow's mirrored strategy is implemented. This strategy enables distributed computing, leveraging multiple processing units in parallel to accelerate the analysis. This approach ensures that computations are executed swiftly, even with large datasets, and allows scalability for future expansions.

Ultimately, the project aims to provide comprehensive insights. This includes a nuanced understanding of customer sentiment towards products, patterns in rating distributions, and an exploration of factors influencing customer satisfaction. By intertwining data preprocessing, sophisticated analytics, and advanced computing techniques, the project endeavors to offer a holistic view of Amazon product reviews, uncovering valuable insights into consumer behavior and product performance.

REFERENCES

Recurrent Neural Networks (RNN):

Reference: Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

Paper Link: Long Short-Term Memory

Long Short-Term Memory networks (LSTM):

Reference: Gers, F. A., Schmidhuber, J., & Cummins, F. (1999). Learning to forget: Continual prediction with LSTM. Neural computation, 12(10), 2451-2471.

Paper Link: Learning to Forget: Continual Prediction with LSTM

Bidirectional LSTMs (BiLSTM):

Reference: Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks, 18(5-6), 602-610.

Paper Link: Framewise phoneme classification with bidirectional LSTM\

BERT (Bidirectional Encoder Representations from Transformers):

Reference: Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Paper Link: BERT: Pre-training of deep bidirectional transformers for language understanding

Justifying recommendations using distantly-labelled reviews and fined-grained aspects
Jianmo Ni, Jiacheng Li, Julian McAuley
Empirical Methods in Natural Language Processing (EMNLP), 2019
pdf